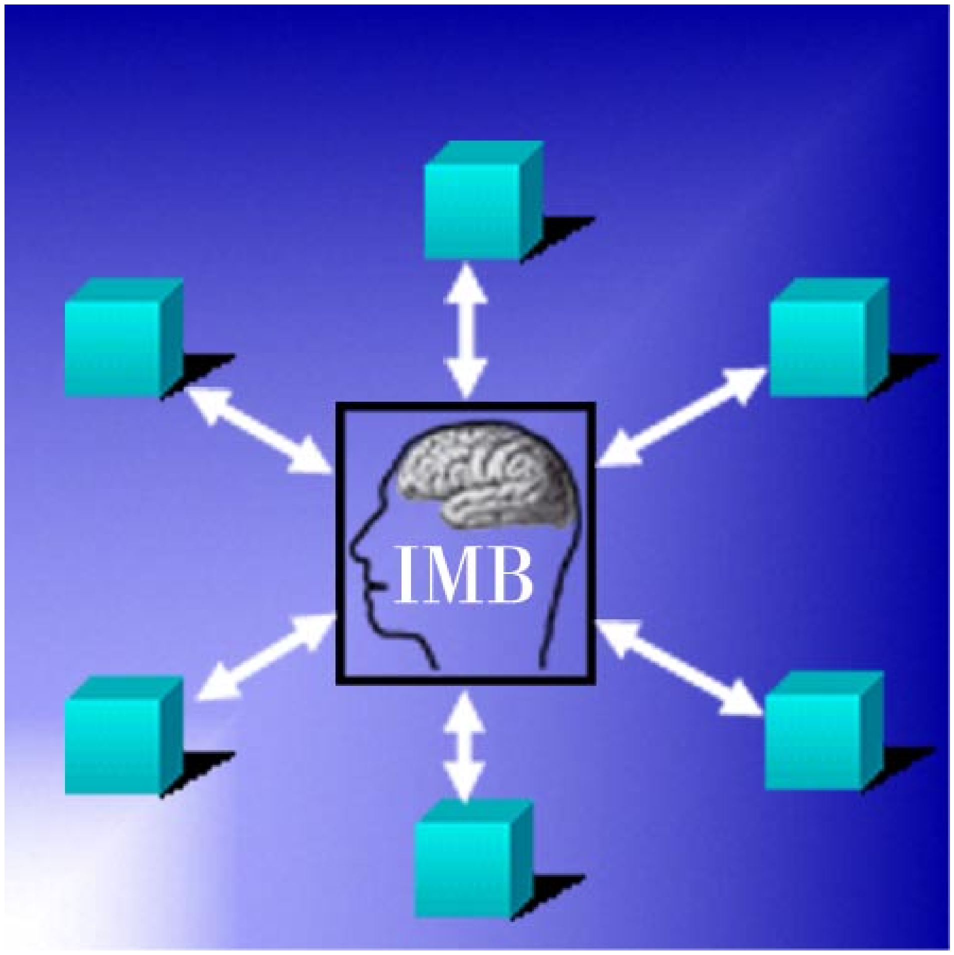
Intelligent Message Broker
for z/OS

IBM

Application Programming Guide

Version 1 Release 0



SC27-1583-00

Intelligent Message Broker
for z/OS

IBM

Application Programming Guide

Version 1 Release 0

SC27-1583-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xi.

First Edition (April 2002)

This edition applies to Version 1, Release 0 of Intelligent Message Broker for z/OS (product number 5799-GPR) and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

Download publications via internet at the following address:

`http://www.ibm.com/software/ad/imb`

Publications are not stocked at the address below.

Please address your comments to:

IBM Global Services
Sortemosevej 21
DK-3450 Alleroed
Denmark

☏ + (45) 45 23 30 00
Fax + (45) 45 23 68 01
E-mail SPOC@dk.ibm.com

Contents

Figures

Tables

Notices

Trademarks and service marks

ACF/VTAM	MQSeries
AIX	MVS/ESA
CICS	OS/2
DB2	OS/390
ExpEDIte	QMF
IBM	RACF
IBM Global Network	VTAM
IMS	WebSphere

Mercator	Mercator Software, Inc.
SAP	SAP AG

Preface

About this book

This book is intended to help you understand how to create and implement applications in Intelligent Message Broker for z/OS (IMB).

This book is split into four parts:

- Part 1, "The MailRoom" on page 1 describes asynchronous support provided by the MailRoom.
- Part 2, "Gateway client/server support" on page 139 describes the synchronous client/server support and explains how this can be used in your applications.
- Part 3, "NPT application design and development" on page 159 describes NPT application design and development under Intelligent Message Broker.
- Part 4, "Common programming APIs" on page 195 describes the application programming interface (API).

Who should read this book

This book is for application designers and developers who need to implement applications using the Intelligent Message Broker infrastructure. This book is also intended for anyone requiring a detailed knowledge of the design principles and applications services provided by Intelligent Message Broker.

Conventions and terminology used in this book

In this book, Intelligent Message Broker for z/OS is referred to as "Intelligent Message Broker" or simply "IMB" where the context makes the meaning clear.

CICS refers CICS Transaction Server for OS/390.

RACF means the Resource Access Control Facility or any other external security manager that provides equivalent function.

Note about version and release numbering

Where the documentation for Intelligent Message Broker - in particular the *Installation Guide* - refers to the actual contents of the product, e.g. dataset names, it will refer to Version 4, Release 5. This is because this product has a history of being distributed internally within IBM for a number of years. Within IBM it has matured to a Version/Release level of 4.5.

Bibliography

Intelligent Message Broker for z/OS books

Intelligent Message Broker for z/OS General Information	GC27-1580
Intelligent Message Broker for z/OS Facilities Guide	SC27-1584
Intelligent Message Broker for z/OS Installation Guide	GC27-1581
Intelligent Message Broker for z/OS Application Programming Guide	SC27-1583
Intelligent Message Broker for z/OS System Administration Guide	SC27-1582
Intelligent Message Broker for z/OS User Administration Guide	SC27-1585

Futher copies of the Intelligent Message Broker for z/OS publications can be downloaded from the product web site:

`http://www.ibm.com/software/ad/imb`

Books from related libraries

CICS TS books

CICS Transaction Server for OS/390 Release Guide	GC34-5352
CICS Transaction Server for OS/390 Installation Guide	GC33-1681
CICS Transaction Server for OS/390 System Definition Guide	SC33-1682
CICS Transaction Server for OS/390 Resource Definition Guide	SC33-1684
CICS Transaction Server for OS/390 Operations and Utilities Guide	SC33-1685
CICS Transaction Server for OS/390 System Programming Reference	SC33-1689
CICS Transaction Server for OS/390 Messages and Codes	GC33-1694
CICS Transaction Server for OS/390 Intercommunication Guide	SC33-1695
CICS Transaction Server for OS/390 Performance Guide	SC33-1699

DataInterchange for MVS books

DataInterchange Messages and Codes	SB34-2000
DataInterchange Programmer's Reference	SB34-2001
DataInterchange Administrator's Guide	SB34-2002

DB2 books

DB2 for OS/390 Administration Guide	SC26-8957
DB2 for OS/390 Application Programming and SQL Guide	SC26-8958
DB2 for OS/390 Command Reference	SC26-8960
DB2 for OS/390 Utility Guide and Reference	SC26-8967

Expedite/CICS books

Customizing and Developing Applications with Expedite/CICS	GC34-3304

IBM EDI Services Information Exchange books

IBM EDI Services Information Exchange Interface Programming Guide	GC34-2222

MQSeries books

MQSeries Planning Guide	GC33-1349
MQSeries for OS/390 System Management Guide	SC34-5374

MVS books

MVS Planning:Global Resource Serialization	GC28-1759

RACF books

OS/390 Security Server (RACF) Administrator's Guide	SC28-1915

Automated Operations Control/MVS books

AOC/MVS Planning and Installation	GC28-1082
AOC/MVS Operations	GC28-1084

ACF/VTAM books

Advanced Communications Function for VTAM Installation and Resource Definition	SC23-0111

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information. If you have any comments about this book or any other IMB documentation:

- Send your comments by email to *SPOC@dk.ibm.com*. Be sure to include the name of the book, the part number of the book, the version of IMB, and, if applicable, the specific location of the text you are commenting on (for example, a page number or table number).
- Mail or fax your comments to the address at the front of this book.

Summary of changes

The major enhancements to this release of Intelligent Message Broker are:

Source Exit
IMB now supports source exits, which makes it easier to implement support for new document types. One new area in this release is XML document routing support. The source scenario will handle incoming documents like this:

- A business document in its original format is passed to IMB. MailRoom will perform a lookup to determine if a source exit should be invoked.
- For MQSeries based scenarios there is an Unpack Exit that should split the received MQ buffer into records.
- For all scenarios (except BATCH and TIE-IMS) there is a Source exit that can build IMB routing information (the M-record) if it is not available in the received document.
- The received document together with the generated M-record is stored in the transport table. The remaining processing is unchanged.
- The MailRoom source exit table holds registrations of exits used by source scenarios. Source exits are either global for a scenario, or specific for a single sender.
- MailRoom supplied source exits will, as default, be used to build the M-record for the DI-EDI, EXP-FILE, SAP and SAP-MQ source scenario. A source exit with XML support is available (see below). Other user written source exits can be written to extend the format support in MailRoom.

XML support
IMB now supports receiving and dynamic routing of XML documents. A XML source exit is available to extract XML elements for building IMB routing information (the M-record). A new panel is available to define which XML elements or attributes should be extracted for each XML document type. A new kernel / destination exit can automatically convert a received XML document to a basic flat file format.

MQSI V1.1 (MVS) reformatter exit
IMB now has an MQSI reformatter exit. In the first version it can format a document to a stream format (records separated with character delimitors), put it on a queue, wait for MQSI to reformat it, get the reformatted stream and finally return it to MailRoom as an updated version of the document.

MailRoom Continuous Receive panel.
A new online function makes it easy to check the status of DataInterchange and Expedite/CICS continuous receives. DI profiles can be stopped and started. It is possible to define a number of profiles as mandatory. A function key can be used to check if the mandatory profiles are active.

Archiving depending on status

Every installation must archive data from the MailRoom DB2 tables in order to remove old data. It is now possible to define the status codes that are applicable for archiving. It is thereby possible to reflect the day to day operational method in the archiving strategy. E.g. FAILed documents are considered open problems and stay in MailRoom until fixed (status changes to FINISH or manually CANCELed). Another principle could be that FAILed documents should be automatically archived.

Schedule process change

A change has been made in the Output Schedule in MailRoom to control the order of sending scheduled envelopes for the same destination. Previously the sending order was in envelope key sequence. Now scheduled envelopes to the same destination are grouped in a single destination envelope and the sending order has been changed to the received order.

Possibility to Mail Text segments

In error situations, when processing EDI like EDIFACT and ANSI X12, a copy of the EDI document is needed for documentation. A new function has been implemented to mail the DataInterchange reports and segments to a specific user Id and node or e-mail address.

Reflow of saved DI audit reports

The text in the DataInterchange audit reports is now automatically reflowed to the width of the panel. It is no longer necessary to scroll right and left to read the audit report.

Segmentation of saved EDIFACT envelope

The saved EDIFACT envelope file is now split in segments to improve the readability.

Handling of a group of envelopes/requests

A new facility is made available to handle more than one envelope using a single group command. Some of the online functions (cancel, fail, finish, resend, restart, sysack) are made available as a group command for multiple envelopes.

Automatic resending of MAIL destination to Internet

We now put documents on the MailRoom XMIT queue if the SMTP server is temporary unavailable.

New MQSeries Batch Read Utility

In addition to the MQSeries Batch Write Utility introduced in last release we now also have a MQSeries Batch Read Utility that can get MailRoom formatted messages from a MQ queue and write them to a file.

SAP R/3 Version 4 improvements

MailRoom now supports sending and receiving SAP R/3 Version 4 IDOCs.

Codepage conversion exit

A new codepage conversion exit is now available as kernel or destination exit. With supplied codepage translation tables it can convert a document from one codepage to another.

Mercator remapping exit improvements

The Mercator exit has been improved with support for selection of profiled or fixed length for input and output (:P or :W parameters to Mercator). Now it is also possible to pass additional command options to Mercator.

Part 1. The MailRoom

The IMB MailRoom can process EDI and other documents asynchronously. The document (application data wrapped with an M-record) is put into the MailRoom by one of the write APIs or send methods, then checked against MailRoom registrations or subscriptions and routed to the final destination. For information about functionality and setup possibilities for the MailRoom, refer to the *System Administration Guide*.

This section covers the programming aspects of using the MailRoom APIs.

These areas are covered:

- Internal structure of the MailRoom
  - The M-record
  - Data formats
  - Multiple or single TS queues
- MailRoom exits
  - Source exits
  - Routing exits
  - SAP global exits
  - Document exits
  - IMB-supplied exits
- The APIs in CICS
  - CICS MailRoom Write API
  - CICS MailRoom Read API
  - CICS MailRoom Acknowledgment API
  - CICS Document Browser API
- The batch utilities
  - Batch MailRoom Write utility
  - Batch MailRoom Read utility
  - Batch MailRoom Write utility using MQSeries.
  - Batch MailRoom Read utility using MQSeries.
- Accessing MailRoom through MQSeries
  - Sending documents
  - Receiving documents
  - Sending acknowledgments
- TIE/IMS Support
  - Sending documents
  - Receiving documents
  - Sending acknowledgments
  - Using TIE with MQSeries.
- TCP/IP APIs (for OS/2, AIX, Windows 95/NT/2000)
  - Generic TCP/IP MailRoom Read and Write Program
  - AIX MailRoom scanner program
  - OS/2 MailRoom Relay File2Tcp and Tcp2File
- APPC APIs (for OS/2, Windows 95/NT/2000)

– Generic APPC MailRoom Read and Write Program

Overview of the MailRoom

Figure 1 shows an overview of the MailRoom components.

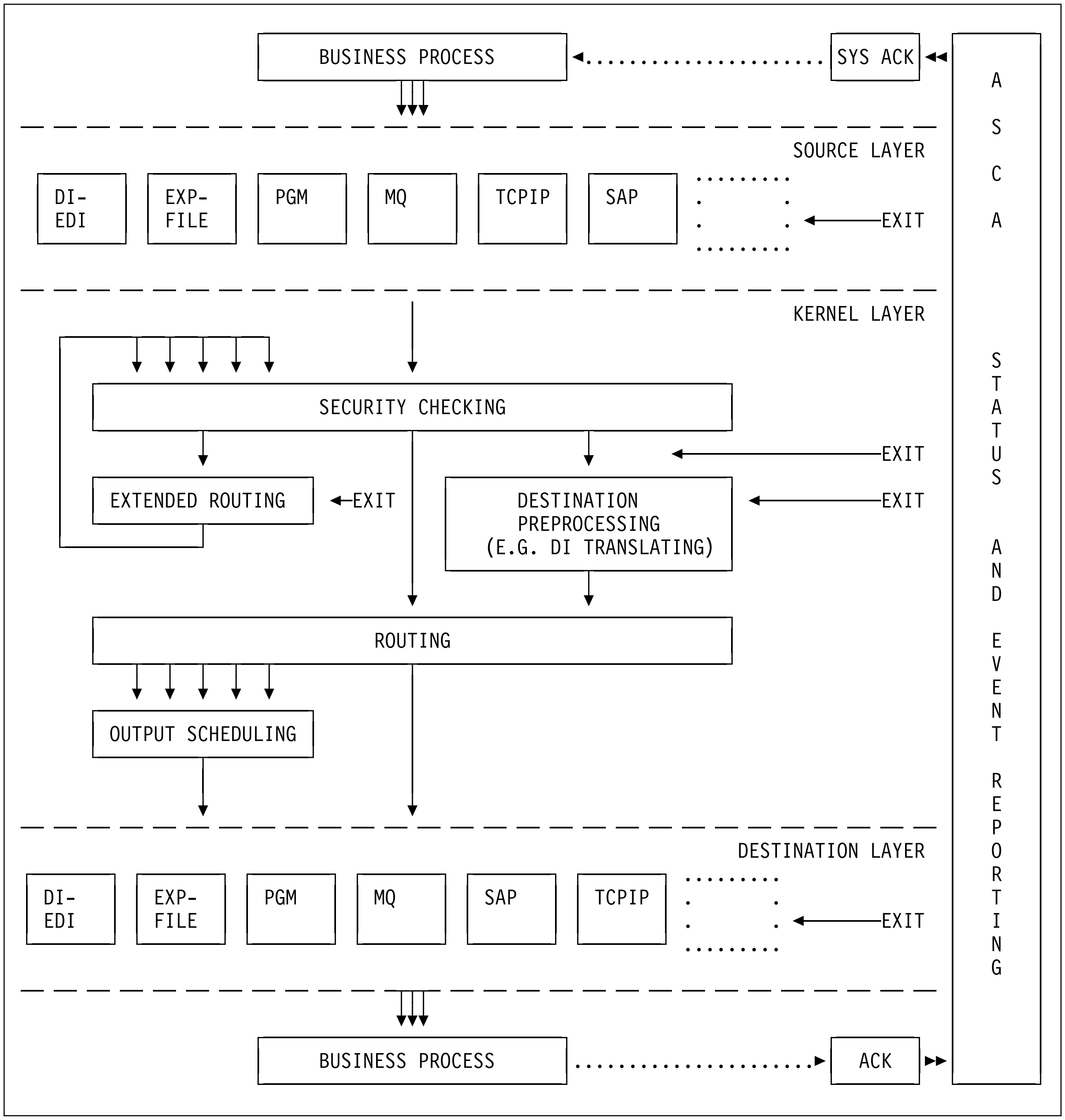


Figure 1. IMB MailRoom component overview

MailRoom layers

The MailRoom is structured into a number of layers, each with a distinct responsibility.

Sending business process

While not part of the MailRoom, the sending business process is the primary initiator of delivery services in MailRoom. It passes a business document to MailRoom and might later receive a System Acknowledgment when processing has completed.

Source Layer

The Source Layer receives the passed document from the sending business process and if necessary converts it to MailRoom format (M- and D- records) via source exits (supplied or user written).

Kernel layer

The Kernel Layer determines the actions to be performed on the received document based on the M-record and MailRoom service and subscription registrations. The document can be inspected or even converted to a different format by a Kernel exit (supplied or user written).

Normally a document is just sent to a single receiver but if necessary it can also be routed to multiple receivers. In Extended Routing it is possible for supplied or user written code to dynamically determine the proper receivers (one or more) based on the document contents or other factors.

When one or more receivers have been found and checked, Routing and Output Scheduling determines if the documents should be sent immediately or at a later point in time.

Destination Layer

At this point the document is ready to be sent, but before doing so, it is possible to let a Destination exit (supplied or user written) inspect or convert the document. Then the document will be passed or sent to the receiver and MailRoom know from the registrations if it should expect acknowledgments from the receiving business process. If sending is not possible due to temporary transmission problems, MailRoom will later automatically initiate a retransmission.

Receiving business process

Finally the document reaches its intended receiver which might generate a network or business acknowledgment after successful processing. The acknowledgment is returned to MailRoom and matched against previous status information about the document and it might also trigger a System Acknowledgment to be returned to the original sending business process.

Monitoring the processing

From the point where the document is received in the source layer until the acknowledgment from the receiving business process has been received, MailRoom monitors the processing and takes *before and after* images of the document. Using monitoring panels, an administrator can watch the progress, view the document before and after an exit has made changes as well as manually initiate certain MailRoom actions like resend, restart, fail, etc..

Chapter 1. MailRoom infrastructure

Document formats

The MailRoom can process various types of business documents, such as invoices and orders. The primary format is M- and D-records, other recognized formats are XML, SAP R/3 IDOC and EDI (EDIFACT and ANSI X12). Even flat files in other formats can be routed through MailRoom if necessary registrations are done or user exits written.

All processing and routing of documents in MailRoom are depending on the MailRoom registrations and identification of sender, receiver and format contained inside the document in an M-record, see "Understanding the M-record" on page 8. If a document is received without an M-record it is necessary for MailRoom to build one. For some source scenarios and well known document types, this can be done automatically in MailRoom supplied source exits. These exits will extract fields from the received document and dynamically build an M-record and if necessary also encapsulate other records in D-records.

If you have other document formats, you must either build the M-record before passing it to IMB or you can write your own source exit. Writing your own source exit is a way to extend MailRoom to understand other document formats, see: "Source exits" on page 29. The received file (or transmission buffer for MQSeries) is passed to the source exit which must return a valid document with M- and D-records.

The document will now be stored in the transport tables as a standard MailRoom document, where the first record is the M-record. The remaining records are data records (first character usually **D**) or special purpose control records (for example DataInterchange C-, E-, I-, and Q-records). One or more documents received within the same syncpoint are considered to be one envelope, which is assigned a unique envelope ID by the MailRoom.

The format support in MailRoom is shown in Table 1 on page 6.

Table 1 (Page 1 of 3). Document formats supported by MailRoom

MailRoom M- and D-records	`MEXTREC00123TPA         TPX            ORDERS`
	`D.....data.....records.....`
	`D.....data.....records.....`
	`D.....data.....records.....`
	`D.....data.....records.....`
	`D.....data.....records.....`
	`D.....data.....records.....`
		Sending:
		All source scenarios except DI-EDI, SAP and SAP-MQ can be used.

		Receiving: All destination scenarios except DI-EDI, SAP and SAP-MQ can be used.
	MailRoom M- and D-records is the native MailRoom format. The received document is stored *As Is* in the MailRoom transport table. The M-record controls the processing and routing in MailRoom. Read about the M-record in "Understanding the M-record" on page 8.	

EDI, EDIFACT, ANSI X.12	`UNA:+./ '`
	`UNB+UN0A:1+MAILBOX1:ZZ+MAILBOX2:ZZ+011015:1310+67'`
	`UNH+128+INVOIC:1:902:UN'`
	`BGM+384:TESTINVOICE+0000323  15+011015'`
	`NAD+SU++John Doe:ADDRES'`
	`NAD+ANY+CITYANYWHERE+DOE:ADDRES ¢2'`
	`UNS+D'`
	`LIN+1++++:1'`
	`LIN+2++++:1'`
	`UNS+S'`
	`TMA+207002'`
	`UNT+11+128'`
	`UNZ+1+67'`
		Sending:
		The DI-EDI source scenario is normally used. An M-record will automatically be built by the global DI-EDI source exit based on information received from DI, see "IMB standard DI-EDI Source Exit KBASXDP" on page 59.

	Example is EDIFACT. ANSI X.12 is also supported.	*Receiving:* The DI-EDI destination scenario is normally used.
	EDI sent to or received from MailRoom via IBM Information Exchange with the DI-EDI scenarios will be translated to or from InHouse by DataInterchange under MailRoom control. The Inhouse format, together with an M-record, is stored in the MailRoom transport table. It is also possible to use other scenarios than DI-EDI to send and receive EDI to/from MailRoom. The EDI could be sent to MailRoom as a flat file (see below) with an M-record built by a source exit. Translation to InHouse format can then be performed with an exit, see "DataInterchange translation exit KBAGXDP" on page 61. The opposite direction is also possible.	

XML	`<Invoice>`
	`   <OrderNumber>AA456</OrderNumber>`
	`   <Date>20010901</Date>`
	`   <Buyer Id="BUYER1">`
	`   <Supplier Id="SUPPLIER1">`
	`   <InvoiceDetails>`
	`      <PartNum>A001</PartNum><Qty>2</Qty><Price>100</Price>`
	`      <PartNum>B033</PartNum><Qty>1</Qty><Price>500</Price>`
	`   </InvoiceDetails>`
	`   <TotalPrice>700</TotalPrice>`
	`</Invoice>`
		Sending:
		All source scenarios except BATCH, DI-EDI, SAP, SAP-MQ and TIE-IMS can be used. It is then necessary to activate an MailRoom supplied source exit to dynamically build an M-record based on defined XML elements, see "IMB XML Processor Source Exit KBASXMP" on page 59.

		Receiving: All destination scenarios except DI-EDI, SAP and SAP-MQ can be used. Use the STRIP-MD function to get the XML document.
	The XML document will be encapsulated in M- and D-records before it is stored in the MailRoom. Read about definition of XML documents for IMB in the *System Administration Guide*. For more information about processing XML documents, refer to "XML document support" on page 15.	

Table 1 (Page 2 of 3). Document formats supported by MailRoom

SAP R/3 IDOC	`EDI_DC   1200000000000009018 30E 69INVOIC012SAPGC0   FIIBMEDI     . . .`
	`E1EDK01  1200000000000009018000001E1EDK01   000000010     GBP`
	`E1EDKA1  1200000000000009018000002E1EDKA1   000000020 RS 1000000010`
	`E1EDK02  1200000000000009018000003E1EDK02   000000020 009VENDOR-INVOIC`
	`E1EDK03  1200000000000009018000004E1EDK03   000000020 01219961215`
		Sending:
		The SAP or SAP-MQ source scenario is normally used. An M-record will automatically be built by a global source exit based on information in the received SAP control record, see “IMB standard SAP & SAP-MQ Source Exit KBASXSP” on page 59.

		Receiving: Normally the SAP or SAP-MQ destination scenario is used. The SAP control record EDI_DC will automatically be built if it is not available in the document.
	The SAP R/3 IDOC will be encapsulated in M- and D-records before it is stored in the MailRoom. If DI-EDI is the source or destination it is possible to use MailRoom supplied document exits to change the length of the record identifier between SAP and DI requirements, see “SAP to DI standard exit KBADXDP” on page 60 and “DI to SAP standard exit KBADXSP” on page 60. For more information about SAP R/3 IDOCs, refer to “SAP R/3 IDOC support” on page 12.	

Flat files	`This is my record oriented flat file, record 1`
	`This is my record oriented flat file, record 2`
	`This is my record oriented flat file, record 3`
	`This is my record oriented flat file, record 4`
	`This is my record oriented flat file, record 5`
	`This is my record oriented flat file, record 6`
	`This is my record oriented flat file, record 7`
	`This is my record oriented flat file, record 8`
	`My last record...`
	Flat files are other record oriented files without MailRoom M-record.
		Sending:
		All source scenarios except BATCH, DI-EDI, SAP, SAP-MQ and TIE-IMS can be used. A source exit must then be used to dynamically build an M-record.
		The EXP-FILE source scenario has a global source exit that will build an M-record based on sender mailbox and message user class, see “IMB standard EXP-FILE Source Exit KBASXFP” on page 59. The sample source exit or a user written one can be used to build an M-record based on file contents or hardcoded values, see “IMB Sample Source Exit KBGXSXP” on page 60.

		Receiving: All destination scenarios except DI-EDI, SAP and SAP-MQ can be used. Use the STRIP-MD function to get the flat file without M- and D-records.
	The flat file will normally be encapsulated in M- and D-records before it is stored in the MailRoom. A record starting with M would be intrepreted as an MailRoom M-record. For more information about writing source exits, refer to “Source exits” on page 31.	

Table 1 (Page 3 of 3). Document formats supported by MailRoom

Other formats	`This is a Very Long Buffer sent via MQSeries.........`	*Sending:* The MQ source scenario can be used to receive other buffer formats than the normal MailRoom MQ buffer format. It is necessary to use a user written source unpack exit to split the buffer into records. MailRoom can also cut records at 1999 byte boundary with MQ default unpack exit.
		Receiving: The MQ destination scenario can be used to send other buffer formats than the normal MailRoom MQ buffer format.
	The received buffer must be cut into apropriate records which will be stored in the MailRoom transport tables.	

Understanding the M-record

The M-record is the MailRoom control record, and it is used to identify and separate documents. Other records must not use the character **M** in position 1.

The M-record exists in a simple and an extended version. The extended M-record is the preferred development method, because it is the only way to use the extended routing and third party routing.

Extended M-record

This is the structure of the MailRoom extended M-record, which is in KBH.R450.PLINCL(KBAMRX00):

```
/* +----------------------------------------------------------------+ */
/* |        -------- INTELLIGENT MESSAGE BROKER (IMB) ---------     | */
/* | (C) Copyright IBM Denmark. 2000.       All Rights Reserved.    | */
/* | (C) Copyright IBM Corp. 2000.          All Rights Reserved.    | */
/* |                                                                | */
/* |                                                                | */
/* +----------------------------------------------------------------+ */

/*   +-----------------------------------------------------------+    */
/*   |       IMB MAILROOM EXTENDED ADDRESSING RECORD             |    */
/*   |       ===========================================         |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBAMRX00                                     |    */
/*   |                                                           |    */
/*   |  LENGTH    : 300 BYTES                                    |    */
/*   |                                                           |    */
/*   |  CONTENTS  : CONTROL INFORMATION ABOUT A DOCUMENT, IS     |    */
/*   |              USED IN IMB MAILROOM                         |    */
/*   |                                                           |    */
/*   |  RELATIONS : BUILT BY EITHER THE CREATOR OF DATA, OR      |    */
/*   |              BY A SOURCE API (DI_EDI AND EXP-FILE)        |    */
/*   |                                                           |    */
/*   +-----------------------------------------------------------+    */
/*                                                                    */
   5 RECID           CHAR(01),    /* CONSTANT 'M'                     */
   5 EYECATCH        CHAR(08),    /* CONSTANT 'EXTREC00'              */
   5 IOPUCTY         CHAR(03),    /* COUNTRY/ORGANIZATION             */
   5 SKBA_TPID_TO    CHAR(35),    /* TO TP (EXTERNAL/INTERNAL)        */
   5 SKBA_TPID_FROM  CHAR(35),    /* FROM TP (EXTERNAL/INTERNAL)      */
   5 SKBA_LAYOUT     CHAR(16),    /* DOCUMENT LAYOUT                  */
   5 SKBA_REF_DATA   CHAR(40),    /* APPLICATION REFERENCE DATA       */
   5 SKBA_REQ_KEY    CHAR(40),    /* MAILROOM REQUEST KEY             */
   5 SKBA_DST_DATA   CHAR(40),    /* DESTINATION SPECIFIC DATA        */
   5 RESERVED        CHAR(82)     /* RESERVED                         */
/*                                                                    */
/*                                                   TOTAL LENGTH 300 */
/*  ==  API  == END OF STRUCTURE KBAMRX00 ==                          */
/*  --------------------------------------------------------------- */
```

Figure 2. Structure of extended M-record

RECID
The record identification.

EYECATCH
An eye-catcher to indicate that this is an extended M-record.

IOPUCTY
Country code.

SKBA_TPID_TO
Trading Partner identification (receiver). The Trading Partner can be either an internal Trading Partner number or an external Trading Partner alias.

SKBA_TPID_FROM
: Trading Partner identification (sender). The Trading Partner can either be an internal Trading Partner number or an external Trading Partner alias.

SKBA_LAYOUT
: MailRoom document layout name.

SKBA_REF_DATA
: Sending application reference field. A free-format field to contain a key known to the sender. The information is available on MailRoom panels.

SKBA_REQ_KEY
: MailRoom request key. Must be blank on all input and writes to MailRoom. On output and reads from MailRoom it contains the MailRoom reference key for the document (request).

  Optionally this field can be used on input to pass a reference MailRoom envelope key to indicate that this new document originates from another envelope. See “Linking envelopes” on page 24 for more information.

SKBA_DST_DATA
: Data specific to the selected destination scenario. Must be blank on all input and writes to MailRoom. In the PGM-CICS scenario it contains the user data from the service or subscription.

RESERVED
: Field reserved for future use. Must be blank.

Simple M-record

This is the structure of the MailRoom simple M-record, which is in KBH.R450.PLINCL(KBAMREC):

```
/* +----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
/* | (C) Copyright IBM Denmark. 1998.       All Rights Reserved.    | */
/* | (C) Copyright IBM Corp. 1998.          All Rights Reserved.    | */
/* |                                                                | */
/* |                                                                | */
/* +----------------------------------------------------------------+ */
/*   +----------------------------------------------------------+    */
/*   |              IMB MAILROOM CONTROL RECORD                  |    */
/*   |              ================================             |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBAMREC                                      |    */
/*   |                                                           |    */
/*   |  LENGTH    : 159 BYTES                                    |    */
/*   |                                                           |    */
/*   |  CONTENTS  : CONTROL INFORMATION ABOUT A DOCUMENT, IS     |    */
/*   |              USED IN IMB MAILROOM                         |    */
/*   |                                                           |    */
/*   |  RELATIONS : BUILT BY EITHER THE CREATOR OF DATA, OR      |    */
/*   |              BY A SOURCE API (DI_EDI AND EXP-FILE)        |    */
/*   |                                                           |    */
/*   +----------------------------------------------------------+    */
/*                                                                    */
   5 RECID          CHAR(01),    /* CONSTANT 'M'                      */
   5 IOPUCTY        CHAR(03),    /* COUNTRY CODE                      */
   5 ICUSPRM        CHAR(09),    /* TRADING PARTNER ID - ACCOUNT      */
   5 SKBA_LAYOUT    CHAR(16),    /* LAYOUT                            */
   5 RESERVED       CHAR(10),    /* FUTURE USE                        */
   5 SKBA_REF_DATA  CHAR(40),    /* APPLICATION REFERENCE DATA        */
   5 SKBA_REQ_KEY   CHAR(40),    /* MAILROOM REQUEST KEY              */
   5 SKBA_DST_DATA  CHAR(40)     /* DESTINATION SPECIFIC DATA         */
/*                                                                    */
/*                                                  TOTAL LENGTH 159 */
/*  ==  API  == END OF STRUCTURE KBAMREC ==                           */
/*  ----------------------------------------------------------------- */
```

Figure 3. Structure of simple M-record

RECID
The record identification.

IOPUCTY
Country code.

ICUSPRM
Internal Trading Partner number. Depending on the subscription type on the service, it is either the sender or the receiver.

SKBA_LAYOUT
MailRoom document layout name.

RESERVED
Field reserved for future use. Must be blank.

SKBA_REF_DATA
: Sending application reference field. A free-format field to contain a key known to the sender. The information is available on MailRoom panels.

SKBA_REQ_KEY
: MailRoom request key. Must be blank on all input and writes to MailRoom. On outputs and reads from MailRoom it will contains the MailRoom reference key for the document (request).

  Optionally this field can be used on input to pass a reference MailRoom envelope key to indicate that this new document originates from another envelope. See "Linking envelopes" on page 24 for more information.

SKBA_DST_DATA
: Data specific to the selected destination scenario. Must be blank on all input and writes to MailRoom. In the PGM-CICS scenario it contains the user data form the service or subscription.

M-record and subscriptions

The M-record in the input data must match a service subscription so the destination can be determined.

```
Input data (extended M-record)
```

```
MEXTREC00123TPA     TPX     ORDERS
D.....data.......
D.....data.......
MEXTREC00123TPB     TPX     ORDERS
D.....data.......
D.....data.......
```

Trading Partner TPX (which can be an alias for a real Trading Partner) sends an envelope containing an ORDER to Trading Partner TPA and to Trading Partner TPB (both can be aliases for real Trading Partner).

TPX must have a send subscription (or be the fixed sender Trading Partner on a service with subscription type *R*) and TPA and TPB must both have a receive subscription to the same service.

SAP R/3 IDOC support

If the destination type is SAP or SAP-MQ, the data must be in the IDOC format as specified by SAP R/3. The MailRoom has support for SAP IDOC Version 2 and 3 for destination type SAP and SAP-MQ, and for SAP-MQ there is also support for SAP IDOC Version 4. The SAP IDOC must be MailRoom encapsulated as described here.

Here is an extract from an SAP IDOC Version 3:

```
EDI_DC    120000000000000901830E 69INVOIC012SAPGC0   FIIBMEDI     . . .
E1EDK01   120000000000009018000001E1EDK01   000000010     GBP
E1EDKA1   120000000000009018000002E1EDKA1   000000020 RS 1000000010
E1EDK02   120000000000009018000003E1EDK02   000000020 009VENDOR-INVOIC
E1EDK03   120000000000009018000004E1EDK03   000000020 01219961215
```

To be able to sucessfully transmit it through MailRoom to SAP R/3 (either through IMB SAP bridge (SAP) or through the MQSeries link for R/3 (SAP-MQ), the IDOC must be placed inside an M-record and D-records as shown here:

```
MEXTREC00120IBMEDI                             1000000010
DEDI_DC    120000000000000901830E 69INVOIC012SAPGC0   FIIBMEDI
DE1EDK01   1200000000000009018000001E1EDK01   000000010     GBP
DE1EDKA1   1200000000000009018000002E1EDKA1   000000020 RS 1000000010
DE1EDK02   1200000000000009018000003E1EDK02   000000020 009VENDOR-INVOIC
DE1EDK03   1200000000000009018000004E1EDK03   000000020 01219961215
```

The SAP Control record EDI_DC or EDI_DC40 is required by SAP, depending on which SAP IDOC Version that is used. The SAP Control record is optionally created by MailRoom, if it is not provided in the sent data.

The EDI_DC record will be primed with values as shown in Table 2 on page 14 for SAP IDOC Version 3. The MailRoom LAYOUT is interpreted in a special way if MailRoom is creating the EDI_DC. The first 8 characters are used to specify the name of the IDOC type (field DOCTYP in EDI_DC) through the mapping table. The next 6 characters specify the EDI message type (field STDMES in EDI_DC).

For SAP IDOC Version 4 the EDI_DC40 will be primed with values as shown in Table 3 on page 15. The mapping between MailRoom LAYOUT and IDOCTYP in the EDI_DC40 record is managed by field *SAP IDoc type* from the Service/Subscription. For use of this field when setting up SAP-MQ source or destination scenarios, refer to *System Administration Guide*.

The sending application can also choose to create the EDI_DC or EDI_DC40 itself and pass it to MailRoom (and thereby specify all possible control values).

The mapping table can be extended to handle other private formats by changing the SAP global naming exit for destination, see “SAP Naming exits” on page 55. Figure 4 on page 14 shows an example of the mapping table.

For SAP sources it is possible to write a source exit that use other principles to build the M-record, see “Source exits” on page 31, or the MailRoom supplied SAP source exit, see “IMB standard SAP & SAP-MQ Source Exit KBASXSP” on page 59 can override individual M-record fields with fixed values.

```
                    char 1-8 of LAYOUT         IDOC type
-------------------+-------------------------+------------
Mapping of          BTCXID01                  BTC_ID01
standard IDOCS      BTCXID02                  BTC_ID02
                    BTCXID03                  BTC_ID03
                    DESXID01                  DES_ID01
                    INVXID01                  INV_ID01
                    ORDXID01                  ORD_ID01
                    WBBXID01                  WBB_ID01
                    WPXEAN01                  WP_EAN01
                    WPXPER01                  WP_PER01
                    WPXPLU01                  WP_PLU01

Mapping of          ZX824XXX                  Z_824
non-standard IDOCs  ZRDXID01                  ZRD_ID01

Other               xxxxxxxx                  xxxxxxxx
```

Figure 4. SAP mapping table

A SAP Inbound Order response in (version 3.0) IDOC format can be addressed with a LAYOUT:
`ORDERS01ORDRSP` (IDOC type ORDERS01 and EDI message type ORDRSP).

The same document in (version 2.x) IDOC format must be named:
`ORDXID01ORDRSP` (IDOC type ORD_ID01 and EDI message type ORDRSP).

Table 2. SAP Control Record priming

SAP Id	Position	Attribute	MR Source	Comment
TABNAM	1-10	Char(10)	Hardcode	EDI_DC
MANDT	11-13	Char(03)	M-record	Current Country Number
STATUS	34-35	Char(02)	Hardcode	50 (IDOC has to be processed)
DOCTYP	36-43	Char(08)	Table	One-to-one mapping table between LAYOUT(1:8) and IDOC-type
DIRECT	44-44	Char(01)	Hardcode	"**2**" meaning SAP Inbound
RCVPOR	45-54	Char(10)	Service/Sub	SAP System Id (receiving system)
RCVPRT	55-56	Char(02)	Service/Sub	SAP Receiver TP Type (SAP TEDST table)
RCVPRN	57-66	Char(10)	M-record	To trading partner from extended M-record
RCVLAD	88-157	Char(70)	DI I-record	Interchange Receiver ID and Group application rec.ID field IRID cols 51-85 and field GRID cols 212-246
STDMES	165-170	Char(06)	M-record	EDI message e.g. **INVOIC** or **850** from LAYOUT(9:6)
MESCOD	171-173	Char(03)	Service/Sub	SAP Logical Msg code
TEST	178-178	Char(01)	DI C-record	column 61 (Test indicator)
SNDPOR	179-188	Char(10)	Hardcode	**MailRoom** left justified
SNDPRT	189-190	Char(02)	Service/Sub	SAP Sender TP Type
SNDPRN	191-200	Char(10)	M-record	From trading partner from extended M-record
SNDLAD	222-291	Char(70)	DI I-record	Logical sender address. Field ISID cols 16-50 and GSID cols 177-211
REFINT	292-305	Char(14)	DI I-record	field IHXCTL cols 2-15. Interchange control number
REFGRP	306-319	Char(14)	DI I-record	field GHXCTL cols 157-170. Group control number
REFMES	320-333	Char(14)	DI I-record	field THXCTL cols 303-316. Transaction control number
ARCKEY	334-403	Char(70)	MR Key	MailRoom reference number left justified

Table 3. SAP Control Record Version 4 priming

SAP Id	Position	Attribute	MR Source	Comment
TABNAM	1-10	Char(10)	Hardcode	EDI_DC40
MANDT	11-13	Char(03)	M-record	Current Country Number
STATUS	34-35	Char(02)	Hardcode	50 (IDOC has to be processed)
DIRECT	36-36	Char(01)	Hardcode	"**2**" meaning SAP Inbound
TEST	39-39	Char(01)	DI C-record	column 61 (Test indicator)
IDOCTYP	40-69	Char(30)	Service/Sub	SAP IDoc type
CIMTYP	70-99	Char(30)	Service/Sub	SAP Extension type
MESTYP	100-129	Char(30)	Service/Sub	SAP SAP Logical message type
MESCOD	130-132	Char(03)	Service/Sub	SAP Logical Msg code
STDMES	143-148	Char(06)	Service/Sub	SAP EDI message type
SNDPOR	149-158	Char(10)	Hardcode	**MailRoom** left justified
SNDPRT	159-160	Char(02)	Service/Sub	SAP Sender TP Type
SNDPRN	163-172	Char(10)	M-record	From trading partner from extended M-record
SNDLAD	194-263	Char(70)	DI I-record	Logical sender address. Field ISID cols 16-50 and GSID cols 177-211
RCVPOR	264-273	Char(10)	Service/Sub	SAP System Id (receiving system)
RCVPRT	274-275	Char(02)	Service/Sub	SAP Receiver TP Type (SAP TEDST table)
RCVPRN	278-287	Char(10)	M-record	To trading partner from extended M-record
RCVLAD	309-378	Char(70)	DI I-record	Interchange Receiver ID and Group application rec.ID field IRID cols 51-85 and field GRID cols 212-246
REFINT	393-406	Char(14)	DI I-record	field IHXCTL cols 2-15. Interchange control number
REFGRP	407-420	Char(14)	DI I-record	field GHXCTL cols 157-170. Group control number
REFMES	421-434	Char(14)	DI I-record	field THXCTL cols 303-316. Transaction control number
ARCKEY	435-504	Char(70)	MR Key	MailRoom reference number left justified

XML document support

XML Documents are widely used in e-Business (B2B) as a replacement for, or alternative to traditional EDI. IMB MailRoom can receive and route XML Documents the same way as flat files with M- and D-records. The XML support in MailRoom is summarized in Table 4 on page 16.

The reader is expected to have some knowledge about XML before reading this section.

Table 4. XML document support. Summary of XML support in MailRoom.

Send format	Description	Receive format	Description
XML	Root element of XML document must be registered and the XML Processor setup as source exit, see: "Receiving XML Documents in MailRoom" on page 16. All MailRoom source scenarios with source exit support can be used.	**XML**	*Supported* This is plain routing without modifications. Remember to remove the MailRoom encapsulation with the function STRIP-MD in the transmission parameters on the service.
		Other	*Possible with user exit* A user written kernel or destination exit is necessary to perform the conversion from XML to another format (corresponding record oriented format). This is not a simple task, since it requires an XML Parser, see: "Reading XML in MailRoom exits." on page 20.
Other	Normal MailRoom setup.	**XML**	*Possible with user exit* A user written kernel or destination must process the received record oriented format and write out the corresponding XML document, which is a straight forward task, see: "Writing XML in MailRoom exits." on page 20.

Receiving XML Documents in MailRoom

The following steps are needed to process XML Documents in MailRoom:

- Setup the IMB XML Processor (source exit)
- Define your XML Documents (how to build M-record)
- Define MailRoom Services and Subscriptions

Using the XML processor source exit

You need to activate the IMB XML processor for your specific use. Normally the processor is not invoked when documents are received and you can therefore not process XML Documents in MailRoom without activating the XML Processor as a source exit. It can either be activated for a single input channel (MQSeries queue, TCP/IP host, etc.) or for all input channels of the same type (same MailRoom source scenario type).

Defining your XML Document in MailRoom

XML Document definitions are used by MailRoom to dynamically assign an M-record for arriving XML Documents. The values for the M-record can either be fixed or determined at runtime based on the contents of the received XML Document.

Identify the Root Element of your XML Document and consider which other elements or attributes should be used when building the M-record.

XML Root Element: The XML Root Element identifies the XML Document type. It is the first normal tag in the XML file. The Root Element is case sensitive. The XML Root Element is the key of the XML Document definition and it must be unique.

Consider an XML Document where the first tag is `<Order>`. MailRoom can process this document if Order is defined as the Root Element.

It should be noted that Order, order and ORDER are different Root Element names because XML element names are case sensitive.

Building the M-record for XML documents: Four fields of the M-record (Country Code, Layout, From & To Trading Partner) are mandatory to have values assigned to and one field (Application Reference) is optional. The values for these fields can either be extracted from the received XML Document at runtime or hardcoded or a combination.

Extracting data from the received XML Document is done by defining a Path to the element or attribute you need to extract.

XML Paths: An XML Path is an absolute or relative addressing method used to identify a specific element or attribute in the XML Document. Addressing an element is done by just giving the element name. Addressing an attribute is done by giving the element name followed by **$** and the attribute name.

The XML processor will always use the first occurrence of an element or attribute matching the XML Path. It can therefore be necessary to further qualify the element by giving the hierarchical structure of the elements separated by ***/***.

Consider the following XML fragment:

```
<Aaaa>
   <Bbbb Ee="CICS">
      <Vvv>Transaction Server</Vvv>
      <Xxx>Customer Information Control System</Xxx>
   </Bbbb>
   <Cccc Ff="APPC">
      <Xxx>Advanced Program-to-Program Communication</Xxx>
   </Cccc>
   <Cccc Ff="TCPIP">
      <Xxx>Transmission Control Protocol/Internet Protocol</Xxx>
   </Cccc>
</Aaaa>
```

The Root Element is `Aaaa`. There is one `Bbbb` element and two `Cccc` elements

```
XML Path         Type  Mode      Expands to value
--------         ----  --------  -----------------
Vvv              Elem  Relative  Transaction Server
Bbbb/Vvv         Elem  Relative  Transaction Server
Cccc/Vvv         Elem  Relative  <not found>
/Aaaa/Bbbb/Vvv   Elem  Absolute  Transaction Server
Xxx              Elem  Relative  Customer Information Control System
Cccc/Xxx         Elem  Relative  Advanced Program-to-Program Communication
/Aaaa/Cccc/Xxx   Elem  Absolute  Advanced Program-to-Program Communication
Bbbb$Ee          Attr  Relative  CICS
/Aaaa/Bbbb$Ee    Attr  Absolute  CICS
Cccc$Ff          Attr  Relative  APPC
/Aaaa/Cccc$Ff    Attr  Absolute  APPC
```

As seen above, element `Xxx` appears in two different elements: `Bbbb` and `Cccc`. For such elements it is recommended to further qualify with the previous element: `Bbbb/Xxx` or `Cccc/Xxx` or the full path: `/Aaaa/Cccc/Xxx`

Restriction: It is currently not possible to extract the last `Xxx` element in second occurrence of `Cccc` (with text: Transmission Control...) because it would require additional input parameters such as:
Give me `Cccc/Xxx` for `Cccc$Ff = "TCPIP"`

Defining MailRoom services and subscriptions

When an XML Document has been successfully processed in the XML Processor it will have a normal M record as first record and all XML lines prefixed with a 'D'.

The setup of services and subscriptions are then normal. The source scenario type should be the same as the method the XML Document originally arrived with.

Which XML Documents are supported

MailRoom needs to browse through the XML Document while building an M-record. For this to work it is necessary to only pass XML Documents already in EBCDIC or to use a MailRoom source scenario with codepage conversion support (e.g. MQSeries, TCP/IP, etc.).

Restriction: Double byte, Unicode and UTF-16 encoded XML Documents are not supported.

Warning: It should also be noted that the encoding attribute inside the XML Document `<?xml version="1.0" encoding="UTF-8"?>` will be invalidated by an ASCII to EBCDIC codepage conversion. The XML Document will then be converted to EBCDIC while it internally still claims to be encoded in ASCII (UTF-8).

This can be a problem if the XML Document originates from an ASCII based computer, and is going to be used on an EBCDIC based computer by a XML parser or receiving program that respect the encoding attribute inside the XML Document. It might be necesasary to instruct the parser to ignore the encoding attribute.

Sample XML Document flow in MailRoom

Fruit Shop Invoice

Joe has a small shop where he sells fruit and other stuff. He has just placed an order of apples, bananas and oranges at Fruit Corporation. Fruit Corporation has delivered the apples and oranges and now wants to send an invoice to Joe for these two line items.

Fruit Corporation is using IMB as its message broker. The invoice is created in the invoicing system and sent to IMB with MQSeries for further delivery via e-mail or other transport methods.

Sample XML

```
<MailRoomXMLInvoice>
   <InvoiceHeader>
      <InvoiceDate>20010901</InvoiceDate>
      <SupplierOrderNumber>AA456</SupplierOrderNumber>
      <BuyerOrderNumber>XXX01ABC99</BuyerOrderNumber>
      <InvoiceCurrency>DKR</InvoiceCurrency>
   </InvoiceHeader>
   <InvoiceParties>
      <Buyer Id="BUYER1">
         <Name1>Joe s Shop</Name1>
         <Address1>Kongevejen 234</Address1>
         <City>Allerord</City>
         <PostalCode>3450</PostalCode>
         <Country>Denmark</Country>
      </Buyer>
      <Supplier Id="SUPPLIER1">
         <Name1>Fruit Corporation</Name1>
         <Address1>Bredgade 321</Address1>
         <City>Copenhagen</City>
         <PostalCode>1325</PostalCode>
         <Country>Denmark</Country>
      </Supplier>
   </InvoiceParties>
   <ListOfInvoiceDetail>
      <InvoiceDetail LineItemNum="1">
         <PartNum>APPLE001</PartNum>
         <ItemDescr>Big box with Granny Smith Apples</ItemDescr>
         <Quantity>5</Quantity>
         <InvoiceUnitPrice>200.75</InvoiceUnitPrice>
      </InvoiceDetail>
      <InvoiceDetail LineItemNum="3">
         <PartNum>ORANG002</PartNum>
         <ItemDescr>Oranges, quality 2, 5 kg. box</ItemDescr>
         <Quantity>1</Quantity>
         <InvoiceUnitPrice>75.00</InvoiceUnitPrice>
      </InvoiceDetail>
   </ListOfInvoiceDetail>
   <InvoiceSummary>
      <SubTotal>1078.75</SubTotal>
      <Tax>
          <TaxPercent>25.0</TaxPercent>
          <Location>Denmark</Location>
          <TaxAmount>269.69</TaxAmount>
          <TaxableAmount>1078.75</TaxableAmount>
      </Tax>
      <Total>1348.44</Total>
   </InvoiceSummary>
</MailRoomXMLInvoice>
```

Extracting values from the XML Document

The Root Element is `Invoice` which we will register in the XML Document definitions table in IMB. This registration must contain specifications on how to build the M-record. By looking at the XML Invoice we might assign the M-record as follows:

M-record field	XML Path	Method	Value
Country Code		hardcoded	123
Layout		hardcoded	XMLINVOICE
From TP	Supplier$Id	evaluated to	SUPPLIER1
To TP	Buyer$Id	evaluated to	BUYER1
Appl Ref.	BuyerOrderNumber	evaluated to	XXX01ABC99

Country Code and Layout will be hardcoded (all XML Documents with Root Element `Invoice` will get these values) and From & To Trading Partner and Application Reference will be taken from the received XML Document.

Flow in MailRoom

The MQSeries Queue used by Fruit Corp. is defined in MailRoom to use the XML Processor as source exit.

The XML processor reads the XML Document and see that the Root Element is Invoice. It then lookup in the XML Document definitions table and finds out how the M-record should be built.

The XML Document is now stored in MailRoom as a normal M-D record Document and processed according to the service and subscription.

Joe might have a subscription, where it is specified that he wants to receive invoices from Fruit Corp. via e-mail.

It is also possible to have a kernel or destination exit that changes the XML Document to another format before it is sent. Some receivers might be able to process the XML while others could need traditional record format.

MailRoom then build an e-mail by stripping off the M-record and removing the D-prefix. The result is a plain XML file that is sent as an e-mail attachment to Joe.

```
<....Sender...>     <................MailRoom............>  <Receiver>
Fruit Corp.  MQSeries     Source exit   Kernel/Dest     e-mail   Joe
              queue      XML processor  Opt. exits   attachment
                         (build M-rec)              (strip M-rec)
```

Processing XML in MailRoom exits.

Existing applications with a need to either send out or receive documents in XML format will need to write a kernel or destination exit to perform conversion between traditional record oriented format and XML format. The generic XML conversion support in IMB is very limited. One very basic conversion exit is available, see “XML Parsing to flat file exit KBAGXXP” on page 64. More advanced conversion must be done in user exits or outside of IMB. The primary reason is that there currently is limited XML support in CICS.

Writing XML in MailRoom exits.

Writing XML is normally not a big problem when the XML format (DTD or schema) is known and all necessary data elements are available in the input record-oriented file. Simply read the input file, e.g. an order and write out the tags corresponding to the fields of the record. The XML file produced in the exit must still be EBCDIC based and placed in records in a TS queue. It is not necessary to encapsulate the XML file in M- and D-records.

Reading XML in MailRoom exits.

Processing XML files normally require an XML Parser. Unfortunately there is very limited XML Parser support in CICS. The XML Toolkit for OS/390 contains parsers for Java (XML4J) and C++ (XML4C), but only the Java version is possible to use in CICS (CICS TS 1.3). Furthermore it must run in a Java Virtual Machine (JVM) (interpreted Java bytecode) while it is not possible to run the Java XML Parser under High Performance Java (HPJ) (compiled Java). Using the Java XML Parser in CICS will therefore have a performance impact due to the high cost of inititating a new JVM for every XML document.

Due to the lack of a high performance XML Parser for CICS, the MailRoom processing of XML documents (extracting fields for the M-record) is using our own implementation of a XML Text Scanner. This tool is also available for usage in other programs like kernel and destination exits, see: “XML Text Scanner, primitive XML Parser—KBHXMLM” on page 243.

See Table 5 on page 21 for a summary of the available options for XML processing inside or outside of IMB.

Table 5. XML conversion options. How can I convert to or from XML

Tool or method	Where	Description
User written code, other to XML	Inside IMB.	User written program writing XML format itself.
User written code, XML to other	Inside IMB.	User written program reading XML format itself.
User written code, XML to other	Inside IMB.	User written program with call to IMB XML Text Scanner, see "XML Text Scanner, primitive XML Parser—KBHXMLM" on page 243. The simple XML conversion exit in IMB is using this method, see "XML Parsing to flat file exit KBAGXXP" on page 64.
User written code, XML to other	Inside IMB.	User written program with call to Java XML Parser. See comments above about performance.
DataInterchange XML support	Outside IMB. in a Batch job	DataInterchange has shipped a PTF with support for XML to/from InHouse format. It runs only in the Batch environment. Support for generic conversion in the CICS environment is announced for DI version 4.
MQSeries Integrator V2	AIX or Windows NT	MQSI V2 has XML conversion support but it currently require transport to and from a distributed platform. IMB has an exit which currently can call MQSI V1 on OS/390, see "MQSI V1 exit KBAGXQP" on page 62.
Note: Intelligent Message Broker Development Group is investigating usage of DataInterchange XML support and/or MQSeries Integrator V2 for implementation of a generic XML conversion exit.		

Document handling in CICS

When the MailRoom receives documents from source scenarios, the documents are first written to CICS temporary storage (TS) queues and later written permanently to the MailRoom Transport Table in DB2. TS queues are also used when interfacing with user applications utilizing the CICS MailRoom Write and Read APIs, and with user-written MailRoom exits.

The physical size limit for a CICS TS queue is 32K records, each with a record length up to 32K bytes. The MailRoom only accepts and handles record lengths up to 2000 bytes, but a document can potentially contain more then 32K records. Therefore large documents will span multiple TS queues.

Multiple TS queues

To handle multiple TS queues an *index* of active queues is needed, or in MailRoom terminology, a List TS Queue (LTSQ), which displays the names of TS queues in use.

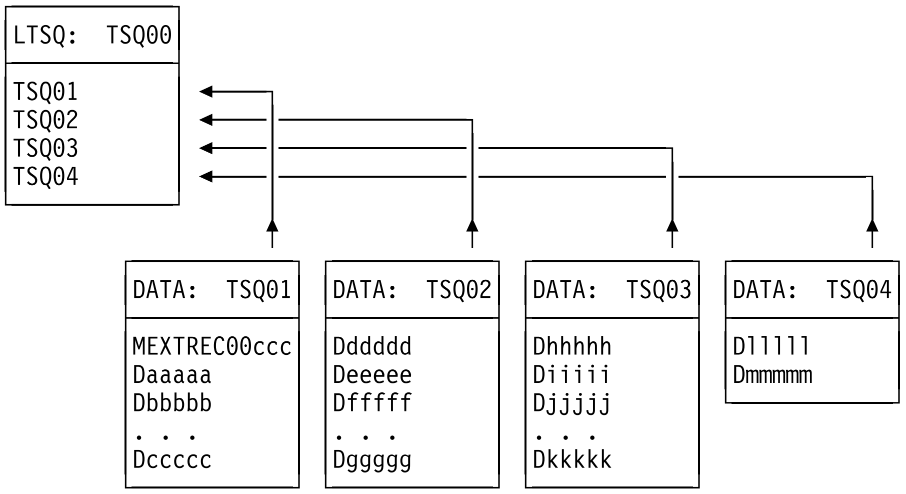


Figure 5. Multiple TS queues

In the example shown in Figure 5, the document spans four TS queues (TSQ01-TSQ04) and the LTSQ (TSQ00) contains four rows containing the TS queue names. The contents of the four data TS queues can be considered as one large document.

To simplify writing size-independent programs, a set of programming routines for PL/1 is provided. With these routines, you can initialize, read, and write documents to or from an LTSQ. The routines and documentation are in the KBHLTSQ *include* member. The sample MailRoom programs also use KBHLTSQ.

Table 6. KBHLTSQ. Summary of available functions.

Function	Description
	To write a large document
LTSQ_INIT_WRITE	Create a new LTSQ and prepare for writing
LTSQ_WRITE_REC	Write a record with length xx to LTSQ
LTSQ_WRITE_RECV	Write a var char string to LTSQ
	To read a large document
LTSQ_INIT_READ	Prepare for reading a LTSQ
LTSQ_READ_REC	Read next record from LTSQ, set length
LTSQ_READ_RECV	Read next record from LTSQ into var char
LTSQ_REWRITE_REC	Rewrite a record with length xx to LTSQ
LTSQ_REWRITE_RECV	Rewrite a var char string to LTSQ
	To position
LTSQ_FIRST	Position to first record in document
LTSQ_LAST	Position to last record in document
LTSQ_GOTO	Position to specified record
	To get size and deletion
LTSQ_SIZE	Calculate size of document
LTSQ_DELETE	Delete LTSQ and associated data TS queues
	To move to and from VSAM file
LTSQ_TO_VSAM	Copy a LTSQ to a VSAM file
LTSQ_FROM_VSAM	Copy a VSAM file to a LTSQ
	Special purpose calls
LTSQ_INIT_WR_ONE	Init a single TS queue for writing via LTSQ_WRITE_REC
LTSQ_INIT_RE_ONE	Init a single TS queue for reading via LTSQ_READ_REC
LTSQ_ADD_DATATSQ	Add a separate data TS queue to a LTSQ

This is in KBH.R450.PLINCL(KBHLTSQ).

All functions use an LTSQ control structure as a parameter, which is in KBH.R450.PLINCL(KBHLTSQS).

Single TS queue—downward compatibility

The method described in “Multiple TS queues” on page 21 provides enough space for large documents, but reading and writing the TS queue becomes more complicated if the document is only 10 lines long. Both the CICS MailRoom Write API, the CICS MailRoom Read API, and the MailRoom document exit API still support single TS queues. This downward compatibility is only available for small documents (less than 32K records contained in one TS queue).

Note: Old programs (accessing a single TS queue) will not operate correctly on large documents. LTSQ methods must be used to process large documents. Either the routines from KBHLTSQ can be used, or programming following the LTSQ principle shown in Table 6.

Implementation in APIs

The CICS MailRoom Write API supports both single TS queue and LTSQ. A switch in the interface structure specifies if the passed TS queue is a single TS queue or an LTSQ. For more information see “CICS MailRoom write API—KBAXWRP” on page 72.

The CICS MailRoom Read API supports both single TS queue and LTSQ. A switch in the interface structure specifies if the application expects the document to be written to a single TS queue (pre&hypyhen.defined TS queue name) or to an LTSQ (dynamically defined TS queue names with specified prefix). For more information see “CICS MailRoom read API—KBAXREP” on page 77.

The MailRoom Document Exit API supports both single TS queue and LTSQ. The exit program always receives the name of the LTSQ. If the document fits into one TS queue, the program also receives the name of a single TS queue. For large documents (more then one TS queue) this field is blank. For more information see “Document exits” on page 42.

The MailRoom Routing Exit API supports only LTSQ. For more information see “Routing exits” on page 56.

The Document Browser API supports only LTSQ. For more information see “CICS MailRoom Document Browser API—KBAXDBP” on page 87.

Moving a large document from LTSQ to a VSAM data set can be necessary when interfacing with other program products. The KBHLTSQ member contains routines to move a LTSQ to and from a VSAM file, but it is still necessary to have a physical VSAM data set, even though usage is temporary. To assist applications in acquiring a VSAM data set from a pool of files and freeing it afterwards, an IMB API can be used. For more information, see “Allocate a VSAM data set from a pool—KBHUVSP” on page 241

Linking envelopes

When new documents are inserted into the MailRoom they are assigned a unique envelope ID. The envelope ID serves as the key to identify:

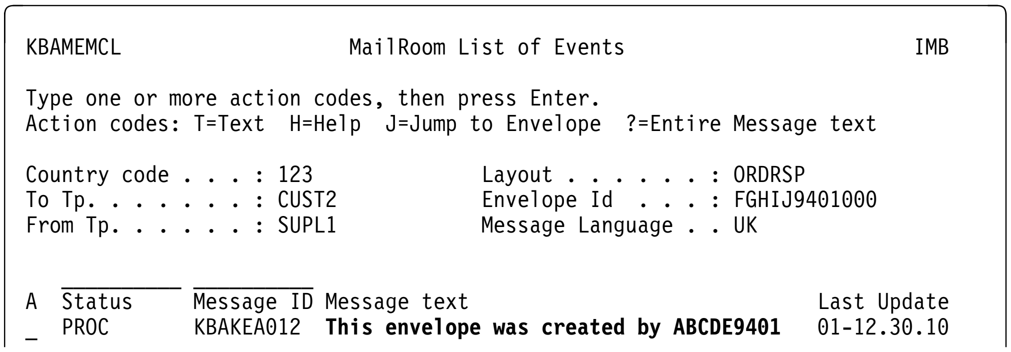
- Documents that arrived together
- Which events occurred for a document request
- Overall status

If one document is processed by an exit, or sent to a destination, and the processing involves creating a new document, the MailRoom can link the two envelope IDs.

With this linking, it is possible to *flip* between the old and the new envelope using the MailRoom status panels. Linking is possible in both directions.

Originating envelope ID for new envelope

While sending a new document into IMB, it is possible to pass the old envelope key in the new M-record. When the information is passed in the correct format, MailRoom will create an extra event containing the originating envelope ID. Using the *Jump* action code, you can display information about the old envelope ID. Jump is possible if the event is recognized as an envelope linking event.



```
KBAMEMCL                   MailRoom List of Events                       IMB

Type one or more action codes, then press Enter.
Action codes: T=Text  H=Help  J=Jump to Envelope  ?=Entire Message text

Country code . . . : 123             Layout . . . . . . : ORDRSP
To Tp. . . . . . . : CUST2           Envelope Id  . . . : FGHIJ9401000
From Tp. . . . . . : SUPL1           Message Language . . UK


A  Status     Message ID Message text                             Last Update
_  PROC       KBAKEA012  This envelope was created by ABCDE9401   01-12.30.10
```

Figure 6. MailRoom List of Events

The information about the *originating envelope ID* should be passed in the M-record in field SKBA_REQ_KEY in this format:

```
M_REC.SKBA_REQ_KEY = 'IMB-ENVKEY'||OLD_ENV_KEY;
```

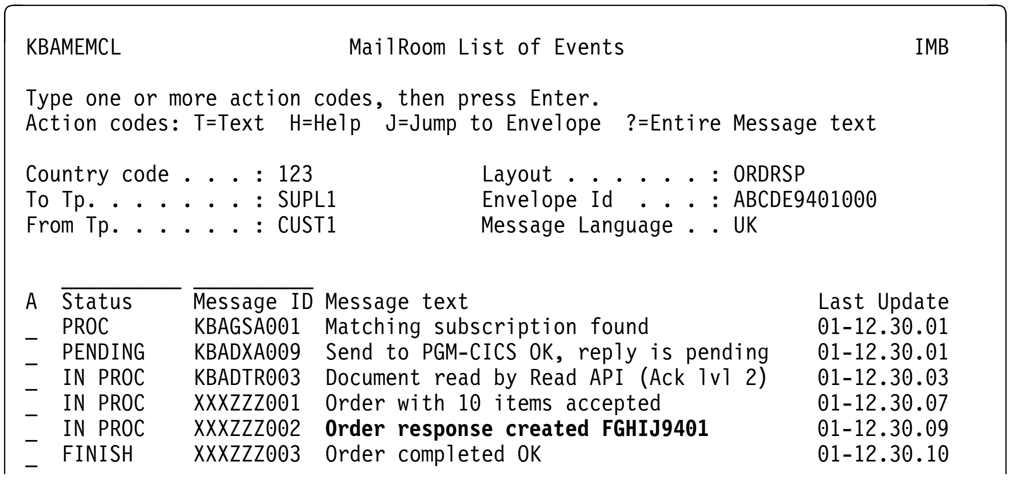
The first 10 bytes contain string *IMB-ENVKEY* and the next 9 bytes contain the originating envelope ID.

If it is not possible to pass the information in the M-record (for example in SAP, where the source scenario has no M-record), you can pass the envelope ID in another field in the document, and then extract the key using a Kernel Exit and create a similar event (see “Creating events that link to another envelope”).

Creating events that link to another envelope

When using the CICS MailRoom Write API to insert a document in MailRoom, the new generated envelope ID is returned. This key can be inserted in an event on the original envelope which has some meaningful text informing that a new document was created with key *xxx*.

Using a normal message ID with special usage of the message variables will mark the event as an envelope linking event, as shown in Figure 7 on page 26.



```
KBAMEMCL                MailRoom List of Events                     IMB

Type one or more action codes, then press Enter.
Action codes: T=Text  H=Help  J=Jump to Envelope  ?=Entire Message text

Country code . . . : 123            Layout . . . . . . : ORDRSP
To Tp. . . . . . . : SUPL1          Envelope Id  . . . : ABCDE9401000
From Tp. . . . . . : CUST1          Message Language . . UK


A  Status     Message ID Message text                          Last Update
_  PROC       KBAGSA001  Matching subscription found           01-12.30.01
_  PENDING    KBADXA009  Send to PGM-CICS OK, reply is pending 01-12.30.01
_  IN PROC    KBADTR003  Document read by Read API (Ack lvl 2) 01-12.30.03
_  IN PROC    XXXZZZ001  Order with 10 items accepted          01-12.30.07
_  IN PROC    XXXZZZ002  Order response created FGHIJ9401      01-12.30.09
_  FINISH     XXXZZZ003  Order completed OK                    01-12.30.10
_
```

Figure 7. Creating events that link to another envelope

The information about the *link to envelope id* should be placed in an event in this format:

```
XXX.SKBH_MSGID  = 'XXXZZZ002';          <<<<< use your own
    ( which has the text: Order response created &2 )
XXX.SKBH_MSGVAR_X(1) = 'IMB-ENVKEY';    <<<<< mandatory
XXX.SKBH_MSGVAR_X(2) = NEW_ENV_KEY;     <<<<< mandatory
```

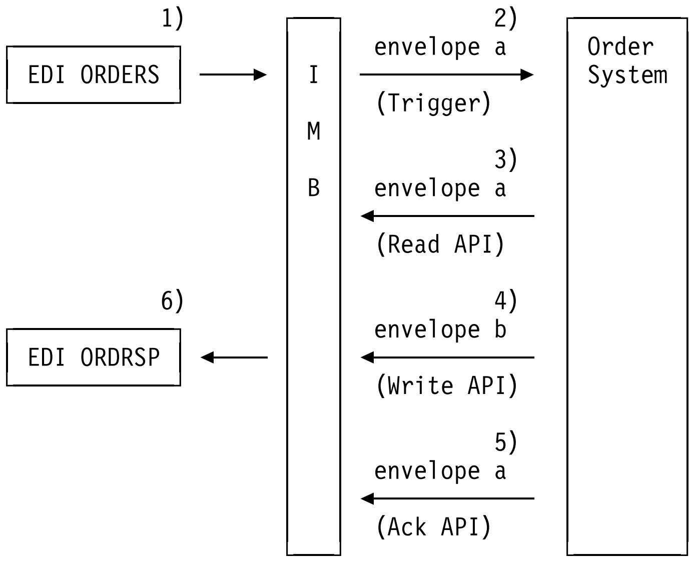
The first message variable must contain the string *IMB-ENVKEY*, and the second message variable must contain the new envelope ID. The message ID can contain any text. New message IDs can be inserted in the IMB Message text table (Error Messages and Event Messages) using option *MSG*.

Such events can be passed either from an exit (if the processing is done in an exit) or using the Acknowledgment API.
For more about:

- Document Exits see “Document exits” on page 42
- Acknowledgments from CICS programs see “CICS MailRoom acknowledgment API—KBAXACP” on page 84
- Acknowledgments from MQSeries see “Sending a Business Acknowledgment to MailRoom using MQSeries” on page 107
- Acknowledgments from TCP/IP see “Sending a Business Acknowledgment to MailRoom using TCP/IP” on page 130
- Acknowledgments from APPC see “Sending a Business Acknowledgment to MailRoom using APPC” on page 138

An example

This feature can be utilized in an order application. A customer sends an order document through DataInterchange (ORDERS) to IMB. The order is processed in a CICS program, and an order response document (ORDRSP) is created. This new document is sent back to the customer through Electronic Data Interchange.



The steps are:

1. An EDI order is sent to IMB and assigned envelope ID (envelope a).
2. The order is routed by IMB to the order system, in this example by triggering a CICS program using a TD queue. Trigger information contains the original envelope ID, envelope 1.
3. The order system uses the CICS MailRoom Read API to receive the order document (envelope 1). The order is validated.
4. An order response is created in a new document. The M-record contains the old envelope ID (envelope a). The CICS MailRoom Write API is called and a new generated envelope ID is returned (envelope b).
5. The order system calls the CICS MailRoom Acknowledgment API to create an event for envelope 1, informing that *Order response created* ***envelope b***
6. The new order response document (envelope b) is routed by IMB, translated with DataInterchange and sent back to the customer.

Chapter 2. MailRoom exits

As shown in the MailRoom overview (see Figure 1 on page 2), you can call a user exit at certain points during MailRoom processing.

Source exits

The MailRoom source layer has two exit points which can be used to reblock or enrich a document before it is saved in MailRoom transport table. Selection of a source exit is depending on the identity of the sender (name/queue/id/etc.) and the source type. An exit is considered *specific* if it is only invoked for a single sender. An exit is considered *global* if it is invoked for all senders of a source scenario.

Unpack Exit	Gets control during the source processing of a document, immediately after data is received and just before a Source Exit. The Unpack Exit should unpack the passed buffer into record format. A Source Unpack Exit can reject a document if the format is unrecognized. The Unpack Exit is available for the MQSeries based scenarios (MQ, SAP-MQ, TIE-MQ) where data is received as one long buffer.
Source Exit	Gets control during the source processing of a document, immediately after data is received. A Source Exit can reject a document if the format is unrecognized.

While the two Source Exits can change data, the intended purpose is only to split the buffer into smaller records and to dynamically build the M-record.

The source exits are defined in the source exit table, see the *System Administration Guide*.

Both user written source exits and MailRoom supplied source exits can be used. The MailRoom supplied source exits can be found in “MailRoom supplied source exits” on page 58. User written source exits must be coded according to the instructions in “Source exits” on page 31.

Document exits

The MailRoom kernel and destination layer also have exit points called document exits which can be used to read and potentially update the document.

Exits are called once per Document (Request) in the Envelope.

Kernel Exit	Gets control during the kernel processing of a document, immediately after the security check has been performed. A Kernel Exit can read or update the document and if necessary *FAIL* or *LOCK* it to stop further processing. The exit can also insert up to five events as MailRoom meta data. These event records can potentially be used to store results of the exit processing. The events can be viewed using standard on-line dialogues.
Destination Exit	Gets control during the destination processing of a document, immediately before further sending or translation. A Destination Exit can read or update the document and if necessary *FAIL* it to stop further processing.

The exit can also insert up to five events as MailRoom meta data. These event records can potentially be used to store results of the exit processing. The events can be viewed using standard on-line dialogues.

Display Exit Gets control during on-line screen display before a specific version of a document has been selected for viewing. Such an exit is selected from a list at display time. The exit can reformat the document to give a better visual presentation of the document on the screen. The result is not saved.

A kernel or destination (document) exit is defined on the MailRoom service or subscription.

Both user written document exits and MailRoom supplied document exits can be used. The MailRoom supplied document exits can be found in “MailRoom supplied document exits” on page 60. User written document exits must be coded according to the instructions in “Document exits” on page 42.

Special purpose exits

Two other exit points are defined:

Routing exits Are an option for MailRoom services defined with Extended Routing. A Routing Exit is used to dynamically select the receivers of a document. Based on its own logic and the contents of the document, it can return a number of *To Trading Partners*, which will receive a copy of the document. The exit must be coded according to the instructions in “Routing exits” on page 56.

SAP naming exit Is called during MailRoom sending of IDOCs to SAP R/3 (for destination type SAP and SAP-MQ). The purpose is to build a SAP control record EDI_DC if one is not already precent in the document. There is one global exit for sending IDOCs, see “SAP Naming exits” on page 55.

Source exits

Source exits are called by the MailRoom source layer every time a document is received and a source exit has been setup for this source scenario on the MailRoom Source Exits panel.

The intended purpose of using source exits is to have some exit points, where an exit can get control of data before data is stored in the MailRoom Transport tables. The exits can be used to unpack data in transmission format into IMB file format, and by this make it possible to transmit data to IMB that is in another format than those supported by IMB. The other main function is to enrich the data with routing information (M-record), if this not already part of data, or to update this routing information.

Note: The exits has the possibility to change the data, but this is not the intended purpose of using source exits.

The source layer has two exit points and also two types of exits that can be used by the source scenarios. The first type of exit *unpack exit* gets control when data is in transmission format. The second type *source exit* gets control when data is in IMB file format. The unpack exit will always get control before the source exit. Table 7 shows the dependence between the source scenario and the type of exit that is available.

Table 7. Source exit availability table

Source scenario type	**Unpack exit**	**Source exit**
APPC	N/A	Yes
BATCH	N/A	N/A
DI-EDI	N/A	Yes
EXP-DOC	N/A	Yes
EXP-FILE	N/A	Yes
MQ	Yes	Yes
PGM	N/A	Yes
SAP	N/A	Yes
SAP-MQ	Yes	Yes
TCPIP	N/A	Yes
TIE-IMS	N/A	N/A
TIE-MQ	Yes	Yes

The use of source exit is controlled by the MailRoom source exit table, which hold all user information to be used by the exits. Every source scenario that supports source exits, will check this table for an exit when data is received. Information from this table is passed to the exit along with the received document, and can be used by the exit in building the IMB file format. The source layer will search the table to find an applicable exit entry. The key to this search is the type of source and the identity of the sender. See Table 8 on page 32 for an explanation of the relation between the type of source and the identity of the sender. The search will attempt to locate an exit in the following order:

- Look for entry matching Source Type, Sender Identity part 1 and Sender Identity part 2. If Sender Identity part 2 is not used for the particulary source type, this step is omitted.
- Look for entry matching Source Type, Sender Identity part 1 and <*>.
- Look for entry matching Source Type and <*>. This will be a global exit.
- If an entry is found, the values for the entry will be used.
- If no entry is found, then no exit will be called and MailRoom is expecting the received data to be in IMB file format with an M-record as the first record.

Table 8. Relation between type of source and Sender Identity

Source scenario type	**Sender Identity part1 (Sender_Id1)**	**Sender Identity part2 (Sender_Id2)**
APPC	Connection/SysID	N/A
DI-EDI	DI Internal TP id	DI Format
EXP-DOC	Msg User Class	Account / Userid
EXP-FILE	Msg User Class	Account / Userid
MQ	MQSeries Queue name	N/A
PGM	CICS Transaction	Userid
SAP	IP Address	N/A
SAP-MQ	MQSeries Queue name	N/A
TCPIP	IP Address	N/A
TIE-MQ	MQSeries Queue name	N/A

Setting up source exits (refer to the *System Administration Guide*).

Unpack exits

The unpack exit is used to unpack the transmission data format into IMB file format. This can be used to send data in an application specific format, and then have a user written exit called to unpack data into IMB file format before data is stored in the MailRoom transport tables.

General conditions for unpack exits

Exits are called to enable specific application data formats to be supported before data is stored in the MailRoom

- Unpack exit splits the data into IMB file format and write document data into a TS queue or LTSQ.
- The exit must setup the return code to indicate the further action for the received data.
- The exit must execute in the same CICS region as the MailRoom

Format

exitname is a CICS Main program, which is LINKed to with this LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('exitname')
                    COMMAREA(KBAXITSU)
                    LENGTH(CSTG(KBAXITSU)); ─────────────────────►◄
```

Parameters

Exits are given access to two types of data using the CICS commarea:

1. MailRoom source exit registration data applicable to this instance.
2. Business application data in transmission format.

Here is the structure of the exit API, which is in KBH.R450.PLINCL(KBAXITSU):

```
/* +----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
/* | (C) Copyright IBM Denmark. 2001.       All Rights Reserved.    | */
/* | (C) Copyright IBM Corp. 2001.          All Rights Reserved.    | */
/* |                                                                | */
/* |                                                                | */
/* +----------------------------------------------------------------+ */

/*  +--------------------------------------------------------------+  */
/*  |             IMB                                              |  */
/*  |             ===============================                  |  */
/*  |                                                              |  */
/*  |  STRUCTURE : KBAXITSU                                        |  */
/*  |                                                              |  */
/*  |  LENGTH    : 01600 BYTES                                     |  */
/*  |                                                              |  */
/*  |  CONTENTS  : API TO SOURCE UNPACK EXIT                       |  */
/*  |                                                              |  */
/*  |  RELATIONS : NONE.                                           |  */
/*  |                                                              |  */
/*  +--------------------------------------------------------------+  */
/*                                                                    */
/*                                                             OFFSET */
/*                                                               ---- */
    3 SKBA_TYPE_SRC      CHAR(08),     /* SOURCE SCENARIO TYPE  0000 */
    3 DOC_DATA,                        /* DOCUMENT DATA         0008 */
     5 DOC_DATA_TYPE     CHAR(03),     /* HOW DATA IS PARSED    0008 */
     5 DOC_DATA_PTR      POINTER,      /* POINTER TO DATA       0011 */
     5 DOC_DATA_LENGTH   FIXED BIN(31),/* LENGTH OF DATA        0015 */
     5 DOC_DATA_TSQ      CHAR(08),     /* TS-QUEUE OR LTSQ      0019 */
     5 DOC_DATA_RESERVE  CHAR(80),     /* FUTURE USE            0027 */
    3 CNTL_DATA,                       /* CONTROL INFORMATION   0107 */
     5 CNTL_DATA_TYPE    CHAR(03),     /* HOW CNTL IS PARSED    0107 */
     5 CNTL_DATA_PTR     POINTER,      /* POINTER TO CNTL INFO  0110 */
     5 CNTL_DATA_LENGTH  FIXED BIN(31),/* LENGTH OF CNTL INFO   0114 */
     5 CNTL_DATA_TSQ     CHAR(08),     /* TS-QUEUE OR LTSQ      0118 */
     5 CNTL_DATA_RESERVE CHAR(80),     /* FUTURE USE            0126 */
    3 ENVIRONMENT_DATA   CHAR(300),    /* ENVIRONMENT SPEC.DATA 0206 */
    3 SKBA_SENDER_ID1    CHAR(64),     /* SENDER IDENTITY       0506 */
    3 SKBA_SENDER_ID2    CHAR(64),     /* SENDER IDENTITY PART2 0570 */
    3 SKBA_UNPACK_PARM   CHAR(40),     /* PARAMETER DATA        0634 */
    3 SKBA_TOTDOC        FIXED BIN(31),/* TOTAL NUMBER OF DOCUM 0674 */
    3 SKBH_EXIT_RC       CHAR(02),     /* RETURN CODE           0678 */
    3 SKBH_MSGID         CHAR(10),     /* MESSAGE ID            0680 */
    3 SKBH_MSGVAR,                     /* MSG VARIABLE          0690 */
     5 SKBH_MSGVAR_X(3)  CHAR(25),     /* 1, 2, 3               0690 */
    3 SKBH_LTSQ          CHAR(08),     /* LTSQ                  0765 */
    3 SKBH_TSQUEUE       CHAR(08),     /* TS-QUEUE              0773 */
    3 SKBA_DOCVERSION    CHAR(30),     /* DOCUMENT VERSION      0781 */
    3 RESERVED           CHAR(789)     /* FUTURE USE            0811 */
/*                                                                   */
/*                                               TOTAL LENGTH 01600 */
/*  ==  END OF STRUCTURE KBAXITSU ==                                 */
/*  -------------------------------------------------------------- */
```

SKBA_TYPE_SRC
Specify the source scenario type that calls the exit.

This is an input field.

DOC_DATA_TYPE
Specify how the data is parsed to the exit:

1. *PTR* - the exit has access to data through a pointer.
2. *TSQ* - the exit has access to data in a single TSQ.
3. *LTS* - the exit has access to data in a LTSQ (see “Multiple TS queues” on page 21).

This is an input field.

DOC_DATA_PTR
Pointer to storage containing data to be unpacked when DOC_DATA_TYPE is *PTR*.

This is an input field.

DOC_DATA_LENGTH
Length of data in storage when DOC_DATA_TYPE is *PTR*.

This is an input field.

DOC_DATA_TSQ
Name of single TS queue or LTSQ holding data to be unpacked when DOC_DATA_TYPE is *TSQ* or *LTS*.

This is an input field.

DOC_DATA_RESERVE
Reserved for future use.

CNTL_DATA_TYPE
Specify how the control data is parsed to the exit:

1. *PTR* - the exit has access to control data through a pointer.
2. *TSQ* - the exit has access to control data in a single TSQ.
3. *LTS* - the exit has access to control data in a LTSQ (see “Multiple TS queues” on page 21).

This is an input field.

CNTL_DATA_PTR

Pointer to storage containing control data when CNTL_DATA_TYPE is *PTR*. The control data might be used by the exit to determine how to unpack the data.

This is an input field.

CNTL_DATA_LENGTH

Length of control data in storage when CNTL_DATA_TYPE is *PTR*.

This is an input field.

CNTL_DATA_TSQ

Name of single TS queue or LTSQ holding control data when CNTL_DATA_TYPE is *TSQ* or *LTS*.

This is an input field.

CNTL_DATA_RESERVE

Reserved for future use.

ENVIRONMENT_DATA

Specific data for this type of source scenario, that can be used by the exit. Data is only available in some source scenarios.

This is an input field.

SKBA_SENDER_ID1

Information about the sender depending on the source scenario type.

This is an input field.

SKBA_SENDER_ID2

Information about the sender depending on the source scenario type. Only available in some source scenarios.

This is an input field.

SKBA_UNPACK_PARM

User parameter data from the source exit table.

This is an input field.

SKBA_TOTDOC

Total number of unpacked documents (number of M-records). This field must be updated by the exit.

This is an output field.

SKBH_EXIT_RC

Exit return code. The value of the return code, determine the further process for the document. The following return codes can be used by the exit:

00 The data has been successfully unpacked into IMB file format and has been placed in a TS queue or LTSQ.

01 The data has been successfully unpacked, but there is no data to be processed by the MailRoom.

08 The exit did not succeed to unpack all data. The document can not be considered as valid, but it has been placed in a TS queue or LTSQ. IMB should build a dummy M-record and the document should be stored in the transport tables. The document will be failed in the MailRoom. The

SKBH_MSGID should be set to indicate why the exit was unable to unpack the data.

16 Severe error has occured. No data has been unpacked and there is no valid data to be stored. Appropiate action will be taken depending on the type of source scenario. The SKBH_MSGID should be set to indicate the error reason.

This is an output field.

SKBH_MSGID

MailRoom Message ID. MailRoom has a standard multi-language message facility that is also available for application use. By storing application messages in the MailRoom message database (DB2), a server can later refer to the message by providing the message number and optionally three variables.

The exit can have a message stored by MailRoom in the error log. This should only be us when SKBH_EXIT_RC is set to '08' or '16'

This is an output field.

SKBH_MSGVAR

Three 25-character variables that can be used to compose application-specific and occurrence-specific event.

This is an output field.

SKBH_LTSQ

The name of the LTSQ holding the unpacked data in IMB file format.

See “Multiple TS queues” on page 21 for more information about reading and writing records in an LTSQ.

This is an output field.

Note: All TS queue names beginning with *KB* are IMB reserved names.

If SKBH_LTSQ is used SKBH_TSQUEUE should be set to blanks.

SKBH_TSQUEUE

The name of the TS queue holding the unpacked data.

If the document is larger than 32K records, this field is be blank, and the only way to access the document is through LTSQ (see “Multiple TS queues” on page 21) and field SKBH_LTSQ.

This is an output field.

Note: All TS queue names beginning with *KB* are IMB reserved names.

If SKBH_TSQUEUE is used SKBH_LTSQ should be set to blanks, otherwise the unpacked data will be processed from the SKBH_LTSQ.

SKBA_DOCVERSION

Version of document. Information is stored along with the document in the transport tables, and is used as information about where this document was created.

This is an output field.

RESERVED

For future use.

Source exits

The source exit is used to build routing information (M-record) if this is not present in the data. The exit can also overwrite existing information. Data to build the M-record is passed to the exit from the source exit table or data can be any information from the received document.

General conditions for source exits

Exits are called to enable specific application data formats to be supported before data is stored in the MailRoom

- Unpack exit splits the data into IMB file format and write document data into a TS queue or LTSQ.
- The exit must setup the return code to indicate the further action for the received data.
- The exit must execute in the same CICS region as the MailRoom

Format

exitname is a CICS Main program, which is LINKed to with this LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('exitname')
                    COMMAREA(KBAXITSX)
                    LENGTH(CSTG(KBAXITSX)); ──────────────────►◄
```

Parameters

Exits are given access to two types of data using the CICS commarea:

1. MailRoom source exit registration data applicable to this instance.
2. Business application data in IMB file format.

Here is the structure of the exit API, which is in KBH.R450.PLINCL(KBAXITSX):

```
/* +-------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------    | */
/* | (C) Copyright IBM Denmark. 2001.      All Rights Reserved.   | */
/* | (C) Copyright IBM Corp. 2001.         All Rights Reserved.   | */
/* |                                                              | */
/* |                                                              | */
/* +-------------------------------------------------------------+ */

/*  +-------------------------------------------------------------+  */
/*  |              IMB                                            |  */
/*  |              ===============================                |  */
/*  |                                                             |  */
/*  |  STRUCTURE : KBAXITSX                                       |  */
/*  |                                                             |  */
/*  |  LENGTH    : 01800 BYTES                                    |  */
/*  |                                                             |  */
/*  |  CONTENTS  : API TO SOURCE EXIT                             |  */
/*  |                                                             |  */
/*  |  RELATIONS : NONE.                                          |  */
/*  |                                                             |  */
/*  +-------------------------------------------------------------+  */
/*                                                                */
/*                                                         OFFSET */
/*                                                         ---- */
    3 SKBA_TYPE_SRC      CHAR(08),     /* SOURCE SCENARIO TYPE  0000 */
    3 DOC_DATA_TYPE      CHAR(03),     /* HOW DATA IS PARSED    0008 */
    3 SKBH_LTSQ          CHAR(08),     /* LTSQ                  0011 */
    3 SKBH_TSQUEUE       CHAR(08),     /* TS-QUEUE              0019 */
    3 ENVIRONMENT_DATA   CHAR(300),    /* ENVIRONMENT SPEC.DATA 0027 */
    3 SKBA_SENDER_ID1    CHAR(64),     /* SENDER IDENTITY       0327 */
    3 SKBA_SENDER_ID2    CHAR(64),     /* SENDER IDENTITY PART2 0391 */
    3 SKBA_SOURCE_PARM   CHAR(40),     /* PARAMETER DATA        0455 */
    3 MREC_INFO,                       /*                       0495 */
     5 IOPUCTY           CHAR(03),     /* COUNTRY CODE          0495 */
     5 SKBA_LAYOUT       CHAR(16),     /* LAYOUT                0498 */
     5 SKBA_TPID_FROM    CHAR(35),     /* TRADING PARTNER FROM  0514 */
     5 SKBA_TPID_TO      CHAR(35),     /* TRADING PARTNER TO    0549 */
     5 SKBA_REF_DATA     CHAR(40),     /* APPL. REFERENCE DATA  0584 */
    3 SKBA_TOTDOC        FIXED BIN(31),/* TOTAL NUMBER OF DOCUM 0624 */
    3 SKBH_EXIT_RC       CHAR(02),     /* RETURN CODE           0628 */
    3 SKBH_MSGID         CHAR(10),     /* MESSAGE ID            0630 */
    3 SKBH_MSGVAR,                     /* MSG VARIABLE          0640 */
     5 SKBH_MSGVAR_X(3)  CHAR(25),     /* 1, 2, 3               0640 */
    3 M_RECORD           CHAR(300),    /* M-RECORD BUILD BY EXIT0715 */
    3 SKBA_DOCVERSION    CHAR(30),     /* DOCUMENT VERSION      1015 */
    3 RESERVED           CHAR(755)     /* FUTURE USE            1045 */
/*                                                                */
/*                                                   TOTAL LENGTH 01800 */
/*  ==  END OF STRUCTURE KBAXITSX ==                              */
/*  ------------------------------------------------------------- */
```

SKBA_TYPE_SRC

Specify the source scenario type that calls the exit.

This is an input field.

DOC_DATA_TYPE

Specify how the data is parsed to the exit:

1. *TSQ* - the exit has access to data in a single TSQ.
2. *LTS* - the exit has access to data in a LTSQ (see "Multiple TS queues" on page 21).

This is an input field.

SKBH_LTSQ

The name of the LTSQ holding the data in IMB file format.

See "Multiple TS queues" on page 21 for more information about reading and writing records in an LTSQ. Data can be updated by the exit or a new LTSQ can be created and returned by the exit.

This is an input/output field.

Note: All TS queue names beginning with *KB* are IMB reserved names.

If a new LTSQ is created, the old LTSQ will be deleted on return from the exit.

If SKBH_LTSQ is used SKBH_TSQUEUE should be set to blanks.

SKBH_TSQUEUE
The name of the TS queue holding the data in IMB file format.

If the document is larger than 32K records, this field is be blank, and the only way to access the document is through LTSQ (see “Multiple TS queues” on page 21) and field SKBH_LTSQ.

This is an input/output field.

Note: All TS queue names beginning with *KB* are IMB reserved names.

If SKBH_TSQUEUE is used SKBH_LTSQ should be set to blanks, otherwise the data will be processed from the SKBH_LTSQ.

ENVIRONMENT_DATA
Specific data for this type of source scenario, that can be used by the exit. Data is only available in some source scenarios.

This is an input field.

SKBA_SENDER_ID1
Information about the sender depending on the source scenario type.

This is an input field.

SKBA_SENDER_ID2
Information about the sender depending on the source scenario type. Only available in some source scenarios.

This is an input field.

SKBA_SOURCE_PARM
User parameter data from the source exit table.

This is an input field.

IOPUCTY Country code from the source exit table.

This is an input field.

SKBA_LAYOUT
Layout from the source exit table.

This is an input field.

SKBA_TPID_FROM
External Trading Partner identification (sender) from the source exit table.

This is an input field.

SKBA_TPID_TO
External Trading Partner identification (receiver) from the source exit table.

This is an input field.

SKBA_REF_DATA
Application reference data from the source exit table.

This is an input field.

SKBA_TOTDOC
Total number of documents (number of M-records).

This is an input field.

SKBH_EXIT_RC

Exit return code. The value of the return code, determine the further process for the document. The following return codes can be used by the exit:

00	The data has been successfully processed and routing information has been build and placed in a TS queue or LTSQ. The M-record *must* be first record in the data.
04	The data has been successfully processed and routing information has been build and placed in *field M_RECORD*. Document data is in TS queue or LTSQ. There must not be any M-records in the TS queue or LTSQ.
05	The data has been successfully processed and routing information has been build and placed in *field M_RECORD*. Document data is in TS queue or LTSQ. Data will be prefixed with 'D' when inserted into MailRoom transport tables. There must not be any M-records in the TS queue or LTSQ.
08	The exit did not succeed to build routing information. The document can not be considered as valid. IMB should build a dummy M-record and the document should be stored in the transport tables. The document will be failed in the MailRoom. The SKBH_MSGID should be set to indicate why the exit was not able to unpack the data.
16	Severe error has occured. No valid data to be stored. Appropiate action will be taken depending on the type of source scenario. The SKBH_MSGID should be set to indicate the error reason.

This is an output field.

SKBH_MSGID

MailRoom Message ID. MailRoom has a standard multi-language message facility that is also available for application use. By storing application messages in the MailRoom message database (DB2), a server can later refer to the message by providing the message number and optionally three variables.

The exit can have a message stored by MailRoom in the error log. This should only be us when SKBH_EXIT_RC is set to '08' or '16'

This is an output field.

SKBH_MSGVAR

Three 25-character variables that can be used to compose application-specific and occurrence-specific event.

This is an output field.

M_RECORD

This field is used *field SKBH_EXIT_RC = '04' or '05'*. The field contains the MailRoom M-record to be used when inserting the document into the MailRoom transport tables. The M-record can be simple or extended, but we will recommend that the extended is used when ever possible. For more information about the M-record (see “Understanding the M-record” on page 8).

This is an output field.

SKBA_DOCVERSION
Version of document. Information is stored along with the document in the transport tables, and is used as information about where this document was created.

This is an output field.

RESERVED
For future use.

Format of TS queue

Application data is passed to the exit in an LTSQ or in a single TS queue. The LTSQ is available in the SKBH_LTSQ field, and a single TS queue is available in the SKBH_TSQUEUE field. The input LTSQ or TS queue will contain the received document, that is expected to be in IMB file format.

To operate on records in an LTSQ, see the available routines in "Multiple TS queues" on page 21.

Many updates to the document by the exit will usually require a new TS queue because CICS has limited change options for TS queues. When creating a new TS queue, the exit must ensure that the name is unique. See "Generate unique TS queue names—KBHUQNP" on page 239 for a method to create a unique TS queue name. Naming of new TS queues is done automatically by the LTSQ routines.

Both original TS queues and new returned TS queues are deleted by MailRoom after use. All temporary queues created by the exit must be deleted before returning to MailRoom.

Examples

Examples of the exits are in:

PLI Sample Source exit: KBH.R450.PLI(KBGXSXM)

A number of general purpose source exits are delivered with IMB, see "MailRoom supplied source exits" on page 58.

Document exits

Kernel exits

Kernel exits are called immediately after security checking has been performed successfully (if rejected by the security function, the exit is not called).

The Kernel exit can read or update the received document. It can also stop further processing of the current document by returning SKBH_EXIT_RC = 08 (current request will FAIL), or it can request the kernel to stop processing all documents in the same source envelope by returning SKBH_EXIT_RC = 16 (all requests and envelope will LOCK).

The LOCK can be used to perform cross-document checking to see if a total counter in the last document matches the sum of all documents.

Destination exits

Destination exits are used to reformat the business data *in-flight*, after it has been read from the MailRoom tables and is in the process of being sent to the destination. For example a Destination exit can be used to insert headers and to handle other formatting items when creating a fax or an e-mail of the business transaction.

The Destination exit can update or read the received document. It can also stop further processing of the current document by returning SKBH_EXIT_RC = 08 (current request will FAIL).

The LOCK function from Kernel exits (returning SKBH_EXIT_RC = 16) is not available in Destination exits.

Display exits

Display exits can be called from the online MailRoom status panel while displaying the contents of a document.

A display exit should be used to enhance the viewing readability of a document for the user. The exit receives the original document from the transport table and reformats it into an appropriate screen display format.

General conditions for exits

Exits are called to enable specific application-related processing to be performed, while the data is the responsibility of the MailRoom.

- Exits are always called at the document (request) level.
- The document (application data) is made available in an LTSQ (see “Multiple TS queues” on page 21), or in a single TS queue if the document is small (less then 32K records).
- Data can optionally be updated by the exit. The update can either take place in the original parsed LTSQ or TS queue, or the exit can create a new LTSQ or TS queue and return it. If a new LTSQ or TS queue is created and returned, MailRoom deletes it after use. The exit must use the process indicator SKBA_PROCESS_IND to indicate that the document has been updated.

- Exits can, optionally, insert up to five events as MailRoom meta data. These event records can potentially be used to store results of the exit processing. The events can be viewed using standard online dialogs and later extracted using SQL queries.
- Exits can call the Document Browser API (see “CICS MailRoom Document Browser API—KBAXDBP” on page 87). This lets you retrieve a document other than the one in progress, provided the exit has access to the key.

Format

exitname is a CICS Main program, which is LINKed to with this LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('exitname')
                    COMMAREA(KBAXIT)
                    LENGTH(CSTG(KBAXIT)); ──────────────────────►◄
```

Exit implementation summary

Exit type	Scenario	MR reg.data	Document	Comment
SRC				See “Source exits” on page 31
KERNEL	All	R/O	R/W	
DEST	DI-EDI	R/O	R/W	Called before DI translation
DEST	All other	R/O	R/W	
DISPLAY	All	R/O	R/W	Modified image not saved

Parameters

Exits are given access to two types of data using the CICS commarea:

1. MailRoom registration data applicable to this instance.
2. Business application data in standard MailRoom format

Here is the structure of the exit API, which is in KBH.R450.PLINCL(KBAXIT):

```
/* +-----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
/* | (C) Copyright IBM Denmark. 2000.      All Rights Reserved.     | */
/* | (C) Copyright IBM Corp. 2000.         All Rights Reserved.     | */
/* |                                                                 | */
/* |                                                                 | */
/* +-----------------------------------------------------------------+ */

/*   +-----------------------------------------------------------+    */
/*   |              IMB MAILROOM                                 |    */
/*   |              ==============================               |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBAXIT                                       |    */
/*   |                                                           |    */
/*   |  LENGTH    : 07000 BYTES                                  |    */
/*   |                                                           |    */
/*   |  CONTENTS  : COMMUNICATION AREA TO MAILROOM EXITS        |    */
/*   |                                                          |    */
/*   |  RELATIONS : INITIALIZED BY MAILROOM WHEN CALLING        |    */
/*   |              EXITS.                                      |    */
/*   |                                                          |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
  5   KBAXIT_REC,                      /* Exit API structure    000 */
   10  SKBH_EXIT_MODE       CHAR(008), /* EXIT TYPE             000 */
   10  SKBH_TSQUEUE         CHAR(008), /* PASSED SINGLE TS QUEUE008 */
   10  SKBH_LTSQ            CHAR(008), /* PASSED LIST TS QUEUE  008 */
   10  KBAXIT_RES1          CHAR(008), /* RESERVED              016 */
   10  SKBA_PROCESS_IND     CHAR(001), /* PROCESS INDICATOR     032 */
   10  SKBH_EVENTS(5),                 /* 5 OPTIONAL EVENT TEXTS033 */
     20  SKBH_MSGID         CHAR(010), /* MESSAGE ID                */
     20  SKBH_MSGVAR,                  /* MSG VARIABLE              */
       30  SKBH_MSGVAR_X(3) CHAR(025), /* 1, 2, 3                   */
   10  SKBH_EXIT_RC         CHAR(002), /* EXIT RETURNCODE       458 */
   10  SKBA_REQ_KEY,                   /* REQ KEY (FOR ACK)         */
    20  SKBA_ENV_DER        CHAR(012), /* DERIVED ENV KEY       460 */
    20  SKBA_DOCSEQNO_NUM   CHAR(010), /* DOC NUMBER            472 */
    20  KBAXIT_RES2         CHAR(018), /* RESERVED              482 */
   10  ICUSPRM_FROM         CHAR(009), /* TP-NO SENDER - INT    500 */
   10  ICUSPRM_TO           CHAR(009), /* TP-NO RECEIVER - INT  509 */
   10  SKBA_TPID_FROM       CHAR(035), /* TP-NO SENDER - EXT    518 */
   10  SKBA_TPID_TO         CHAR(035), /* TP-NO RECEIVER - EXT  553 */
   10  SKBA_LAYOUT          CHAR(016), /* LAYOUT OF DATA        588 */
   10  SKBA_REF_DATA        CHAR(040), /* APPLICATION REF.DATA  604 */
   10  SKBA_REF_DATA2       CHAR(040), /* APPLICATION REF.DATA2 644 */
   10  IOPUCTY              CHAR(003), /* COUNTRY/ORG CODE      684 */
   10  SKBH_EXIT_PARM       CHAR(040), /* EXIT PARMETERS        687 */
   10  KBAXIT_RES3          CHAR(073), /* RESERVED              727 */
   10  SKBH_ATTR(100)       CHAR(016), /* ATTRIBUTE NAME        800 */
   10  SKBH_VALUE(100)      CHAR(040), /* ATTRIBUTE VALUE      2400 */
   10  KBAXIT_RES4          CHAR(600)  /* RESERVED             6400 */
/*                                                                   */
/*                                                TOTAL LENGTH 07000 */
/*  ==  END OF STRUCTURE KBAXIT   ==                                 */
/*                                                                   */
```

SKBH_EXIT_MODE

Eye-catcher indicating from where the exit is called. Currently the contents are *KERNEL*, *DEST*, or *DISPLAY*.

SKBH_TSQUEUE

The name of the TS queue holding the document. The exit can update the document, either by returning a new TS queue name, or by updating the contents of the original TS queue. The exit must use the process indicator SKBA_PROCESS_IND to indicate that the document has been updated. MailRoom will then save a new version of the document in the transport table.

If the document is larger than 32K records, this field is be blank, and the only way to access the document is through LTSQ (see "Multiple TS queues" on page 21) and field SKBH_LTSQ.

Note: All TS queue names beginning with *KB* are IMB reserved names. Original TS queue and returned TS queue are deleted after use.

SKBH_LTSQ
The name of the LTSQ holding the data. The exit can update the document, either by returning a new LTSQ, or by updating the contents of the document in the original LTSQ. The exit must use the process indicator SKBA_PROCESS_IND to indicate that the document has been updated. MailRoom will then save a new version of the document in the transport table.

See “Multiple TS queues” on page 21 for more information about reading and writing records in an LTSQ.

Note: All TS queue names beginning with *KB* are IMB reserved names. Original TS queue and returned TS queue are deleted after use.

SKBA_PROCESS_IND
Initialized to blank. Valid return values are *R* or blank for *Read*, and *U* for Updated. This field must be set with the value *U* if the exit updated the document.

SKBH_MSGID
MailRoom Message ID. MailRoom has a standard multi-language message facility that is also available for application use. By storing application messages in the MailRoom message database (DB2), a server can later refer to the message by providing the message number and optionally three variables.

In this case an exit can have a message stored by MailRoom as an *event* in the MailRoom repository if a message number is returned here. Up to five events can be specified for insertion by MailRoom.

If this field is blank, and SKBH_MSGVAR is also blank, the MailRoom will insert an event using the generic messages KBAGXA011 and KBAGXA012, depending on the contents of SKBA_PROCESS_IND.

If SKBH_MSGID is blank and MSGVAR is not blank, KBAGXA010 is used with the passed message varables:

```
KBAGXA010   UK  I  &1&2&3
KBAGXA011   UK  I  Document browsed by exitname
KBAGXA012   UK  I  Document updated by exitname
```

SKBH_MSGVAR
Three 25—character variables that can be used to compose application-specific and occurrence-specific events.

SKBH_EXIT_RC
Exit return code. Can be used by the exit to request MailRoom to FAIL the document. Valid values *00* for OK and *08* for FAIL. A kernel exit can also return *16* to LOCK the full source envelope.

Document a fail request with one or more events (SKBH_MSGID) explaining the error situation.

SKBA_ENV_DER
MailRoom internal key for the derived envelope (part of SKBH_REQ_KEY).

SKBA_DOCSEQNO_NUM
MailRoom internal document sequence number within the envelope. (part of SKBH_REQ_KEY)

ICUSPRM_FROM
MailRoom internal Trading Partner account number (sender).

ICUSPRM_TO
MailRoom internal Trading Partnerr account number (receiver).

SKBA_TPID_FROM
External Trading Partner identification (sender).

SKBA_TPID_TO
External Trading Partner identification (receiver).

SKBA_LAYOUT
MailRoom document layout name.

SKBA_REF_DATA
40 bytes of application (reference) data. Optionally containing reference data set up by the sender.

SKBA_REF_DATA2
40 bytes of application (reference) data. Optionally containing reference data returned by the recipient business application server, using the MailRoom acknowledgement API.

This field can also be updated by the exit, but its use must be coordinated with the server processing.

If used, MailRoom will insert the field in the MailRoom Request table that can be browsed on the Request Status online panels or used by off-line application access.

IOPUCTY Country code.

SKBH_EXIT_PARM
Exit parameters as defined on the service panel. Can be used to start the same exit with different parameters.

SKBH_ATTR
Array with up to 100 field names of the fields that have been used to register data for this particular service or subscription. The set differs depending on the service.

SKBH_VALUE
The corresponding array with up to 100 field contents.

Format of TS queue

Application data is passed to the exit in an LTSQ or in a single TS queue. The LTSQ is available in the SKBH_LTSQ field, and a single TS queue is available in the SKBH_TSQUEUE field (which will be blank for documents with more than 32K rows).

To operate on records in an LTSQ, see the available routines in “Multiple TS queues” on page 21.

The input LTSQ or TS queue will contain one document (request).

The format of the M-record and the selection of rows depends upon the exit type and the definitions of the service or subscription. A kernel exit sees the unchanged document, whereas the format of the document given to a destination exit is controlled by the registrations on the service or subscription.

Many updates to the document by the exit will usually require a new TS queue because CICS has limited change options for TS queues. When creating a new TS queue, the exit must ensure that the name is unique. See "Generate unique TS queue names—KBHUQNP" on page 239 for a method to create a unique TS queue name. Naming of new TS queues is done automatically by the LTSQ routines.

Both original TS queues and new returned TS queues are deleted by MailRoom after use. All temporary queues created by the exit must be deleted before returning to MailRoom.

Examples

Examples of the exits are in:

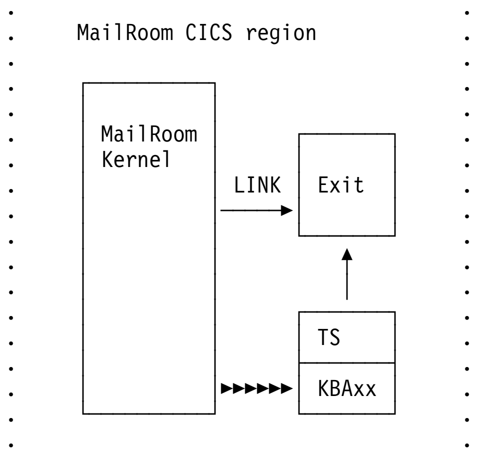
PLI Sample Kernel and Destination exit: KBH.R450.PLI(KBGXITM)

PLI Sample Display exit: KBH.R450.PLI(KBGXIDM)

A number of general purpose document exits are delivered with IMB, see "MailRoom supplied document exits" on page 60.

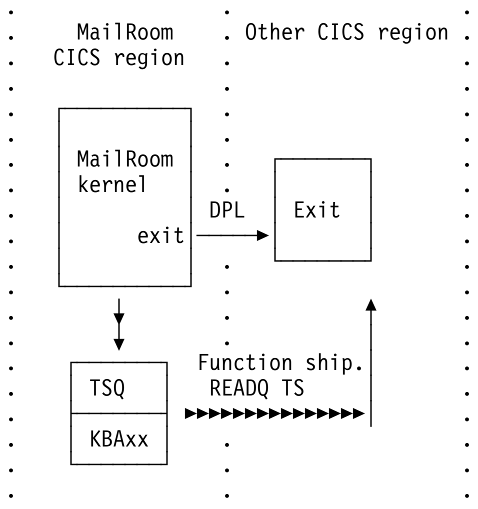
Exit and MailRoom in same the CICS region

The exit usually runs in the same CICS as the MailRoom and all resources are defined as local.



Exit and MailRoom in different CICS regions

Using CICS DPL (Distributed Program Link), an exit can run in another region. The TS Queue with the business data is written locally to a queue (beginning KBA), and the exit should be set up to read that particular TS Queue prefix KBA remotely.



Processing

All exits will be able to stop further processing for the transaction. By returning a return code of 08, it instructs the MailRoom to *FAIL* the transaction according to standard MailRoom processing.

A kernel exit can also decide to stop the processing of a full source envelope by returning a return code of 16. This will LOCK all the requests and the envelope and prohibit later restart.

An exit also has the possibility to update the 40 byte SKBA_REF_DATA2 field that MailRoom will store together with other control data for this transaction.

If the exit is linked locally and is accessing DB2, the required DBRMs must be included in the IMB DB2 Plan *KBHPLAN*.

If the exit is linked through DPL, the necessary DB2 access must be established in the remote CICS.

MailRoom Service and Subscription attributes

Table 9 shows the *potential* contents of the SKBA_ATTR(100) and SKBH_VALUE(100) arrays in the KBAXIT structure. The actual contents differs depending on the active service and subscriptions.

The resulting arrays are quite dynamic. It is necessary to scan for a particular field name in the ATTR array and - if found - its current value can be read from the corresponding entry in the VALUE array.

Table 9 (Page 1 of 6). Service definition table

Field name	Length	Description
ACKLVL	001	Acknowledgement level: X, 0, 1 or 2

Table 9 (Page 2 of 6). Service definition table

Field name	Length	Description
DST_ACCNT_INFO	040	Accounting info, Receiver
DST_BECAPL	016	Server Application name in BEC
DST_BECLOC	008	Server Location (node name)
DST_CIMTYP	030	SAP Extension type
DST_CPCONV	008	EBCDIC-ASCII translation table name
DST_DIEX_CP	008	DI exit: EBCDIC-ASCII translation table name
DST_DIEX_DILOG	008	DI exit: Log (Application id)
DST_DIEX_DITRAN	040	DI exit: Translate options
DST_DIEX_IH_TSQ	008	DI exit: Name of IHF TS queue
DST_DIEX_TYPE	016	DI exit: Translation mode. Values: EDI-IHF, IHF-EDI or MAIL-EDI-IHF
DST_DIFORMAT	016	DI override format name
DST_DILOG	008	DI log (Application id)
DST_DISEND	040	DI send options
DST_DITEST	001	DI test indicator
DST_DITRAN	040	DI translate options
DST_EXIT_PARM1	040	Parameters to destination exit
DST_EXIT1	008	Name of destination exit
DST_FAXATT	040	Receiver FAX attention name
DST_FAXNO_L	040	Receiver Fax-number (See comment below for layout.)
DST_IDOCTYP	030	SAP IDOC type (Version 4)
DST_IEADR_L	040	Receiver IE Account user ID (See comment below for layout.)
DST_M_REC_FORMAT	008	Format of returned M-record. Values: (' ': Simple M-record)|('EXTREC00': extended M-records)
DST_MAIL_CODEPG	008	EBCDIC-ASCII translation table name
DST_MAIL_FILEID	040	Name of file to send
DST_MAIL_SENDER	040	Override address of sender
DST_MAIL_SUBJECT	040	Subject of mail or fax
DST_MAILADR_L	040	Receiver VM Node user ID (See comment below for layout.)
DST_MAILFNC	008	How should mail be sent. Values: NOTE or FILE (or default value blank for FILE)
DST_MASK_FNC	008	Read API mask function. What records have been passed? Values: ('CD': C+D-records)|('MD': M+D records)|('RAW': Everything including DI Info.records)|('STRIP-MD': Only D records without the D)|(Blank: ⇒'MD')
DST_MERC_ADDOPT	040	Mercator exit: Additional Command options

Table 9 (Page 3 of 6). Service definition table

Field name	**Length**	**Description**
DST_MERC_DEBUG	009	Mercator exit: Envelope id to debug
DST_MERC_IORATIO	008	Mercator exit: Input/Outupt ratio
DST_MERC_MAP	007	Mercator exit: Map name
DST_MERC_PAGING	008	Mercator exit: Paging
DST_MERC_PARM1	040	Mercator exit: User parameter
DST_MERC_PROFI	001	Mercator exit: Prof./Fixed lgth(I)
DST_MERC_PROFO	001	Mercator exit: Prof./Fixed lgth(O)
DST_MERC_STATUS	001	Mercator exit: Request status doc
DST_MERC_VALIDAT	004	Mercator exit: Validation type
DST_MERC_VAR	008	Mercator exit: Which variant
DST_MERC_WRKFILE	001	Mercator exit: Workfiles in memory
DST_MESCOD	003	SAP Logical message code
DST_MESTYP	030	SAP Logical message type
DST_MQ_DATA_TYPE	008	Format of data send via MQSeries
DST_MQ_MGR	048	Receiver MQSeries Queue Manager
DST_MQ_QUEUE	048	Receiver MQSeries Queue name
DST_MQ_REPLY	048	Reply MQSeries Queue name
DST_MQEX_MQ_MGR	044	MQSI exit: MQ Queue manager
DST_MQEX_MQ_PUTQ	044	MQSI exit: MQ Queue to PUT on
DST_MQEX_MQ_GETQ	044	MQSI exit: MQ Queue to GET from
DST_MQEX_DELAY	010	MQSI exit: Max wait time for GET
DST_MQEX_MQSI_AP	016	MQSI exit: MQSI Appl. Group
DST_MQEX_MQSI_MS	016	MQSI exit: MQSI Message Type
DST_MQEX_MQSI_RL	003	MQSI exit: MQSI Reload Rule Set
DST_MQEX_PROT_DL	002	MQSI exit: Transport delimitors
DST_MQEX_PROT_IG	003	MQSI exit: Ignore delim. in data
DST_MSGUCLS	008	IE Message User Class
DST_RCVPOR	010	SAP System Id
DST_RCVPRT	002	SAP Receiver partner type
DST_REQID	016	DI Requestor id. Used by DI to select which mailbox is used to send from.
DST_SAPVERS	008	SAP R/3 Version
DST_SNDPRT	002	SAP Sender partner type
DST_STDMES	006	SAP EDI message type
DST_SUEXIT_PARMn	040	Super exit: Exit 1–5 parameters
DST_SUEXITn	008	Super exit: Exit 1–5 name
DST_SYSID	004	CICS Sysid for remote APPC system
DST_TCPADR_L	040	Destination TCP/IP address (See comment below for layout.)

Table 9 (Page 4 of 6). Service definition table

Field name	Length	Description
DST_TCPPORT	005	TCP/IP port number to send to. Recommended value: '01812'
DST_TDQUEUE	004	Name of TDQueue that triggers the PGM-CICS scenario
DST_TIEAPPL	004	TIE application name (re. IMS server)
DST_TIEBTX	008	TIE BTX name (re. IMS server)
DST_TPPGM	008	Remote APPC TP Program
DST_USERFLD	040	Free format data sent to PGM-CICS
ENABLE	001	Service status 0: Disabled | 1: Enabled
EXT_EXIT1	008	Name of routing exit for exit based routing
IOPUCTY	003	Service sponsor country/organization code
IPRAIDY	008	Service name
ISYSIDY	004	System/Application id. Used to group services together on administration panels
KNL_CPEX_CP	008	Codepage exit: EBCDIC-ASCII translation table name
KNL_DIEX_CP	008	DI exit: EBCDIC-ASCII translation table name
KNL_DIEX_DILOG	008	DI exit: Log (Application id)
KNL_DIEX_DITRAN	040	DI exit: Translate options
KNL_DIEX_IH_TSQ	008	DI exit: Name of IHF TS queue
KNL_DIEX_TYPE	016	DI exit: Translation mode. Values: EDI-IHF, IHF-EDI or MAIL-EDI-IHF
KNL_MERC_ADDOPT	040	Mercator exit: Additional Command options
KNL_MERC_DEBUG	009	Mercator exit: Envelope id to debug
KNL_MERC_IORATIO	008	Mercator exit: Input/Outupt ratio
KNL_MERC_MAP	007	Mercator exit: Map name
KNL_MERC_PAGING	008	Mercator exit: Paging
KNL_MERC_PARM1	040	Mercator exit: User parmameter
KNL_MERC_PROFI	001	Mercator exit: Prof./Fixed lgth(I)
KNL_MERC_PROFO	001	Mercator exit: Prof./Fixed lgth(O)
KNL_MERC_STATUS	001	Mercator exit: Request status doc
KNL_MERC_VALIDAT	004	Mercator exit: Validation type
KNL_MERC_VAR	008	Mercator exit: Which variant
KNL_MERC_WRKFILE	001	Mercator exit: Workfiles in memory
KNL_MQEX_MQ_MGR	044	MQSI exit: MQ Queue manager
KNL_MQEX_MQ_PUTQ	044	MQSI exit: MQ Queue to PUT on
KNL_MQEX_MQ_GETQ	044	MQSI exit: MQ Queue to GET from
KNL_MQEX_DELAY	010	MQSI exit: Max wait time for GET
KNL_MQEX_MQSI_AP	016	MQSI exit: MQSI Appl. Group
KNL_MQEX_MQSI_MS	016	MQSI exit: MQSI Message Type

Table 9 (Page 5 of 6). Service definition table

Field name	Length	Description
KNL_MQEX_MQSI_RL	003	MQSI exit: MQSI Reload Rule Set
KNL_MQEX_PROT_DL	002	MQSI exit: Transport delimitors
KNL_MQEX_PROT_IG	003	MQSI exit: Ignore delim. in data
KNL_SUEXIT_PARMn	040	Super exit: Exit 1–5 parameters
KNL_SUEXITn	008	Super exit: Exit 1–5 name
KRNL_EXIT_PARM1	040	Parameters to kernel exit
KRNL_EXIT1	008	Name of kernel exit
OVDTIME	010	Overdue time. Number of minutes
POCELADR_L	040	POC electronic address. Used to send alert messages (See comment below for layout.)
POCLANG	002	POC preferred language (of message)
POCNAME	040	POC name. Text; not folded.
POCTPID	009	POC Account number / Fixed TP service
SKBA_ARCHIVE_PRO	008	Name of archive profile
SKBA_INP_SCHDUL	020	Input schedule
SKBA_LAYOUT	016	Layout name. Layout field from M-rec
SKBA_LOG_LEVEL	008	Extended logging level. Normal logging: ' '|Extended logging: 'VERBOSE'
SKBA_NPRADSC	040	Service description. Text
SKBA_NSUBENA	001	Subscription status. Disabled:'0' | Enabled: '1'
SKBA_PRIORITY	008	Priority level. Normal:' ' | High: 'HIGH'
SKBA_RULESET	016	Name of ruleset for rule based routing
SKBA_TYPE_DST	008	Destination scenario type. E.g. 'DI-EDI'. For full list see prompt on registration panel.
SKBA_TYPE_SRC	008	Source scenario type. E.g. 'TCPIP' For full list see prompt on registration panel.
SKBA_TYPE_SUB	001	Subscription type. Values: SID. "S"ource for TP-to-Sponsor. "D"est for Sponsor-to-TP.
SKBA_TYPE_SYSACK	008	System Acknowledgement type Values: 'MAIL' | 'MQ' | 'PGM-CICS' | 'TIE-IMS' | 'TIE-MQ' | ' ' (blank, for no SYSACK)
SKBA_TYPE_X_ROUT	008	Type of extended routing Values: 'EXIT' | 'RULE' | 'FANNING'
SKBH_SCHKEY	008	Schedule key. Name of schedule
SRC_ACCNT_INFO	040	Sender account information. Text
SRC_IDOCTYP	030	SAP IDOC type (Version 4)
SRC_IEADR_L	040	Sender IE Account user ID. NB! Only available if Source coming from IE. (See comment below for layout.)
SRC_MQ_QUEUE	048	Name of sending MQSeries queue
SRC_MSGUCLS	008	Sender Message User Class

Table 9 (Page 6 of 6). Service definition table

Field name	Length	Description
SRC_SAPDSCP	008	Name of codepage (SAP status flow)
SRC_SAPDSPO	005	TCP/IP port to send to. Recommended value: '01812'
SRC_TCPADR_L	040	Sender TCP/IP address (See comment below for layout.)
SUBCOMM	040	Subscription comment. Free text.
SYA_BECAPL	016	Sysack BEC application name
SYA_BECLOC	008	Sysack BEC location (node)
SYA_DATA_TYPE	008	Sysack Format of MQ data
SYA_M_REC_FORMAT	008	Sysack Format M-record
SYA_MAILADR_L	040	Sysack MAIL address
SYA_MQ_MGR	048	Sysack MQ Manager
SYA_MQ_QUEUE	048	Sysack MQ Queue
SYA_SYSA_FNC	008	Sysack What data to send
SYA_TDQUEUE	004	Sysack CICS TD Queue
SYA_TIEAPPL	004	Sysack TIE application name
SYA_TIEBTX	008	Sysack TIE BTX name
SYA_USERFLD	040	Sysack CICS user field

Layout of electronic address fields passed

Some of the MailRoom Service and Subscription attributes contains electronic addresses. The contents of the VALUE field depends on the electronic address type. Table 10 shows the attributes and Table 11 on page 54 shows the type-dependent overlay.

Table 10 (Page 1 of 2). Attributes with electronic addresses

Field name	Contents of VALUE
DST_FAXNO_L	CHAR 02 MASK: Should be FX CHAR 38 Layout below
DST_IEADR_L	CHAR 02 MASK: Should be IE CHAR 38 Layout below
DST_MAILADR_L	CHAR 02 MASK: Should be HO|VM|ME|TL|LI|IN CHAR 38 Layout below
DST_TCPADR_L	CHAR 02 MASK: Should be TC CHAR 38 Layout below
POCELADR_L	CHAR 02 MASK: Should be TC|VM|ME|TL|LI|IN CHAR 38 Layout below
SRC_IEADR_L	CHAR 02 MASK: Should be IE CHAR 38 Layout below
SRC_TCPADR_L	CHAR 02 MASK: Should be TC CHAR 38 Layout below

Table 10 (Page 2 of 2). Attributes with electronic addresses

Field name	Contents of VALUE
SYA_MAILADR_L	CHAR 02 MASK: Should be HO|VM|ME|TL|LI|IN CHAR 38 Layout below

Table 11. Type dependent overlay for electronic addresses

Type	Layout
MASK = FX	Remaining 38 char as follows: CHAR 08 Country telephone code CHAR 16 Telephone number CHAR 14 Fill
MASK = IE	Remaining 38 char as follows: CHAR 08 Network CHAR 08 Account CHAR 08 user ID CHAR 01 Record format (0|1|2) CHAR 13 Fill
MASK = HO|VM|ME	Remaining 38 char as follows: CHAR 08 user ID CHAR 08 Nodeid CHAR 22 Fill
MASK = TC	Remaining 38 char as follows: CHAR 38 Hostname or IP Address xxx.xxx.xxx.xxx
MASK = TL	Remaining 38 char as follows: CHAR 08 FORUM name CHAR 08 FORUM type CHAR 08 Nickname (from WC table) CHAR 14 Fill
MASK = IN	Remaining 38 char as follows: CHAR 38 Internet e-mail address
MASK = LI	Remaining 38 char as follows: CHAR 08 List name CHAR 30 Fill

SAP Naming exits

When sending IDOCs to SAP R/3, MailRoom optionally creates an EDI_DC record if it does not already exist in the passed data. This task is performed by the SAP Destination Scenario Global Naming exit KBADSXP, which is called before any destination exit.

In previous releases of IMB there were a similar exit used when sending SAP R/3 IDOCs to MailRoom: *SAP Source Scenario Global Naming exit KBASSXP*.

This exit is no longer available since the logic is now performed in the MailRoom supplied source exit KBASXSP, see “IMB standard SAP & SAP-MQ Source Exit KBASXSP” on page 59.

SAP Destination Scenario Global Naming exit KBADSXP

This is a global exit, that can be customized to perform special processing to suit the needs of an organisation.

The source of KBADSXP is in KBH.R450.PLI(KBADSXM).

Note: Because the exit might be changed by any new release or fix package of IMB, it is necessary to perform any local modifications again after an upgrade.

The exit gets access to the document through an LTSQ (see “Multiple TS queues” on page 21) and a structure containing MailRoom registration information from the service or subscription.

On return from the exit, the document contains a valid IDOC with a complete EDI_DC record. The IDOC must be MailRoom encapsulated (see “SAP R/3 IDOC support” on page 12).

Current logic

In the standard version, the exit scans the document and captures certain records from DataInterchange, and updates the document with a complete EDI_DC record. If the EDI_DC was already provided by the sender only certain fields in EDI_DC will be updated (ARCKEY). Refer to Table 2 on page 14 for information about the complete priming of the SAP R/3 EDI_DC record.

Routing exits

When Subscription Type has been specified as *X* (Extended Routing) on the service, it is possible to select Exit as the routing mechanism. A Routing exit gets control during MailRoom Kernel processing, and can dynamically select which receiver Trading Partners will receive a copy of the document.

The Kernel then checks the subscription and sends a copy of the document to the destination defined on the receive subscription (through the returned receiver Trading Partner).

Format

exitname is a CICS Main program, which is LINKed to using this LINK syntax:

LINK Syntax (PL/1)

```
►►── EXEC CICS LINK PROGRAM('exitname')
                    COMMAREA(KBAXROUT)
                    LENGTH(CSTG(KBAXROUT)); ──────────────────►◄
```

Parameters

Here is the structure of the interface, which is in KBH.R450.PLINCL(KBAXROUT):

```
 /* +----------------------------------------------------------------+ */
 /* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
 /* | (C) COPYRIGHT IBM DENMARK. 1998.       ALL RIGHTS RESERVED.   | */
 /* | (C) COPYRIGHT IBM CORP. 2000.          ALL RIGHTS RESERVED.   | */
 /* |                                                                | */
 /* |                                                                | */
 /* +----------------------------------------------------------------+ */

 /*   +-----------------------------------------------------------+    */
 /*   |         STRUCTURE FOR ROUTING EXIT/MODULES                 |    */
 /*   |         ==================================                 |    */
 /*   |                                                            |    */
 /*   |  STRUCTURE : KBAXROUT                                      |    */
 /*   |                                                            |    */
 /*   |  LENGTH    : 1000 BYTES                                    |    */
 /*   |                                                            |    */
 /*   |  CONTENTS  : COMMUNICATION AREA TO RULE DECISION MODULES   |    */
 /*   |                                                            |    */
 /*   |  RELATIONS :                                               |    */
 /*   |                                                            |    */
 /*   +-----------------------------------------------------------+    */
 /*                                                                    */
    3 KBAXROUT_REC                          ,/*                        */
      5 SKBH_FNC              CHAR(008)     ,/* FUNCTION               */
      5 SKBH_LTSQ             CHAR(008)     ,/* LTSQ, INPUT            */
      5 SKBA_TPID_TO          CHAR(035)     ,/* TP-NO RECEIVER - EXT   */
      5 SKBA_REQ_KEY                        ,/* REQ KEY (FOR ACK)      */
       7 SKBA_ENV_DER         CHAR(012)     ,/* DERIVED ENV KEY        */
       7 SKBA_DOCSEQNO_NUM    CHAR(010)     ,/* DOC NUMBER             */
       7 KBAXROUT_RES1        CHAR(018)     ,/* RESERVED               */
      5 IOPUCTY               CHAR(003)     ,/* COUNTRY CODE           */
      5 ISYSIDY               CHAR(004)     ,/* APPLICATION ID         */
      5 IPRAIDY               CHAR(008)     ,/* SERVICE NAME           */
      5 SKBA_LAYOUT         CHAR(016)    ,/* LAYOUT OF DATA          */
      5 SKBA_RULESET        CHAR(016)    ,/* RULESET NAME            */
      5 SKBH_EVENTS (5)                  ,/* UP TO 5 EVENTS          */
        7 SKBH_MSGID        CHAR(010)    ,/* MESSAGE ID              */
        7 SKBH_MSGVAR                    ,/* MESSAGE VARIABLE        */
          9 SKBH_MSGVAR_X(3) CHAR(025)   ,/* MSGVAR 1,2,3            */
      5 SKBA_LOG_LEVEL      CHAR(008)    ,/* LOG LEVEL               */
      5 SKBA_TSQ_TPLIST     CHAR(008)    ,/* RETURNED LIST OF TPS    */
      5 SKBA_TSQ_TRACE      CHAR(008)    ,/* RETURNED TRACE IF ENABL */
      5 SKBA_RC             CHAR(002)    ,/* RETURN CODE             */
      5 KBAXROUT_RES2       CHAR(411)     /* FUTURE USED             */
 /*                                                                  */
 /*                                                TOTAL LENGTH 1000 */
 /*  ==  API  == END OF STRUCTURE KBAXROUT ==                        */
 /*  --------------------------------------------------------------- */
 /*                                                                  */
```

SKBH_FNC
Function code, currently not used. Blank value.

SKBH_LTSQ
Name of the LTSQ holding the names of TS queues containing the document that routing is dependent on. See “Multiple TS queues” on page 21 for more information about reading records from an LTSQ.

SKBA_TPID_TO
Receiver Trading Partner number as originally specified in the passed M-record. The exit can choose to respect or ignore this Trading Partner as a receiver Trading Partner.

SKBA_ENV_DER
MailRoom internal key for the derived envelope (part of SKBH_REQ_KEY).

SKBA_DOCSEQNO_NUM
MailRoom internal document sequence number within the envelope (part of SKBH_REQ_KEY).

IOPUCTY
Country code.

ISYSIDY
Application ID.

IPRAIDY
Service name.

SKBA_LAYOUT
MailRoom document layout name.

SKBA_RULESET
Name of ruleset. Blank for Routing exits.

SKBH_MSGID
MailRoom Message ID. MailRoom has a standard multi-language message facility that is also available for application use. By storing application messages in the MailRoom message database (DB2), a server can later refer to the message by providing the message number and optionally three variables.

In this case an exit can have a message stored by MailRoom as an *event* in the MailRoom repository if a message number is returned here. Up to five events can be specified for insertion by MailRoom.

SKBH_MSGVAR
Three 25-character variables that can be used to compose application-specific and occurrence-specific events.

SKBA_LOG_LEVEL
MailRoom logging level. Can either be blank for normal logging or *VERBOSE* for extended logging. If extended logging is active, the Routing exit can choose to produce an execution trace (free format text about the processing) in a TS queue (see SKBA_TSQ_TRACE) and up to five event messages. With normal logging, the exit should not produce an execution trace and limits the events to a minimum.

SKBA_TSQ_TPLIST
Name of the TS queue that contains a list of receiver Trading Partner IDs on return from the exit. Each Trading Partner must occupy a separate 35-character record. The Kernel sorts the TS queue and removes duplicates.

SKBA_TSQ_TRACE
If extended logging is specified (see SKBA_LOG_LEVEL), this field contains the name of a TS queue that can be used to produce an execution trace. The trace will be available as event text on the MailRoom status panel.

SKBA_RC
Exit return code. Can be used by the exit to request MailRoom to FAIL the document. Valid values are *00* for OK and *08* for FAIL.

Examples

An example of a routing exit is in KBH.R450.PLI(KBGXRTM).

MailRoom supplied source exits

Some standard source exits are supplied with IMB. These exits are described here.

IMB standard MQ, SAP-MQ & TIE-MQ Unpack Exit KBASUMP

For MQSeries based source scenarios the default is to use this exit to unpack the received MQ buffer into records in one or more TS queues. The exit can decode the MQSeries Link for SAP R/3 buffer format and the IMB MailRoom buffer format.

Usage Defined as MailRoom global Unpack Exit for the following source scenarios: MQ, SAP-MQ, TIE-MQ

Override None, done in Source Exit

Reuse You can reuse this exit as a specific exit via source exit administration panels.

IMB standard DI-EDI Source Exit KBASXDP

For the DI-EDI source scenario the default is to use this exit to build an M-record based on the received Inhouse format and used response program.

Usage Defined as MailRoom global Source Exit for the DI-EDI source scenario.

Override Individual fields in the M-record can be overridden.

Reuse You can reuse this exit as a specific exit via source exit administration panels.

IMB standard EXP-FILE Source Exit KBASXFP

For the EXP-FILE source scenario the default is to use this exit to build an M-record based on the message user class and the sender mailbox. Normally a simple M-record is built, but by overriding the To TP it is possible to change this to an extended M-record.

Usage Defined as MailRoom global Source Exit for the EXP-FILE source scenario.

Override Individual fields in the M-record can be overridden.

Reuse You can reuse this exit as a specific exit via source exit administration panels.

IMB XML Processor Source Exit KBASXMP

This exit can parse a received XML document and extract elements or attributes to be used in the M-record. The XML root element must be defined in the MailRoom XML Document definition table. Documents with M- and D-records are also accepted transparently.

Usage Not defined as global Source Exit for any source scenario. The exit must explicitly be activated for a scenario.

Override Individual fields in the M-record can be overridden.

Reuse Can be used as both global or specific source exit via source exit administration panels. You can thereby enhance e.g. the TCPIP source scenario to both accept M- and D-records as well as XML documents or you can enable the XML support for a single MQ queue.

IMB standard SAP & SAP-MQ Source Exit KBASXSP

For the SAP and SAP-MQ source scenarios the default is to use this exit to build an M-record based on the SAP EDI_DC control record in the received IDOC.

Usage Defined as MailRoom global Source Exit for the SAP and SAP-MQ source scenarios.

Override Individual fields in the M-record can be overridden.

Reuse You can reuse this exit as a specific exit via source exit administration panels.

IMB Sample Source Exit KBGXSXP

This exit is a sample program that can be used as a starting point when writing a new source exit.

Usage	Normally not used.
Override	Individual fields in the M-record can be overridden.
Reuse	You can use this exit to simply hardcode an M-record via source exit administration panels.

MailRoom supplied document exits

Some standard exits are supplied with IMB. These exits are described here.

SAP to DI standard exit KBADXDP

Destination exit used to convert SAP IDOCs to DI in-house format when destination is DI-EDI.

The exit removes underscores from the record name and expands the key from 10 to 16 characters. The exit also maps the external Trading Partner id to the C-record, which is used by DataInterchange to match the Trading Partner.

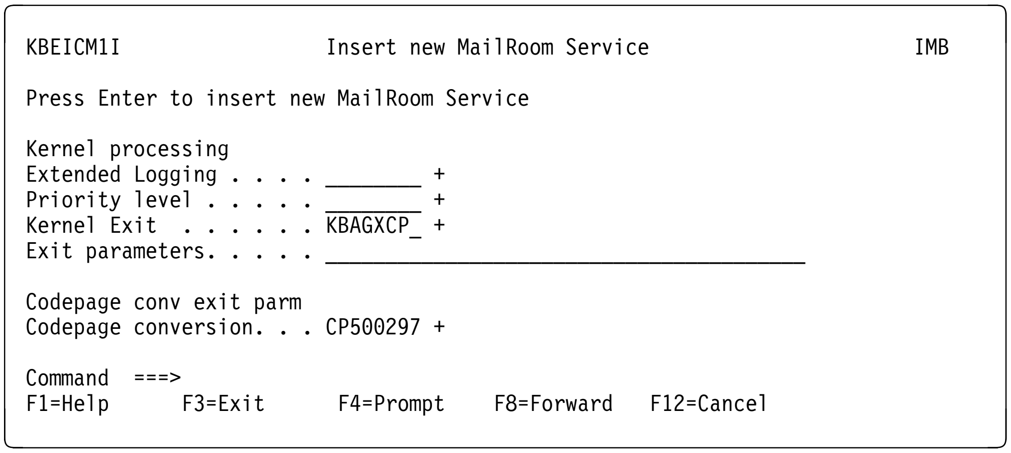
DI to SAP standard exit KBADXSP

Destination exit used to convert DI inhouse format to SAP IDOC when destination is SAP or SAP-MQ.

The exit inserts underscores in the record name where needed and decreases the key from 16 to 10 characters. The exit also populates all records with the SAP client number. The exit will not modify the *EDI_DC* record.

Codepage conversion exit KBAGXCP

Kernel or destination exit used to perform codepage conversion of a document. The conversion will normally be from one EBCDIC codepage to another EBCDIC codepage. This can be necessary if a document is sent to MailRoom in one EBCDIC codepage and the receiver needs it in another EBCDIC codepage. The example shown here uses the kernel exit. Use the Insert New MailRoom Service panel to set up the codepage conversion exit parameters. Figure 8 on page 61 shows the relevant section of the panel.



```
KBEICM1I                Insert new MailRoom Service                      IMB

Press Enter to insert new MailRoom Service

Kernel processing
Extended Logging . . . . ________ +
Priority level . . . . . ________ +
Kernel Exit  . . . . . . KBAGXCP_ +
Exit parameters. . . . . ________________________________________

Codepage conv exit parm
Codepage conversion. . . CP500297 +

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F8=Forward   F12=Cancel
```

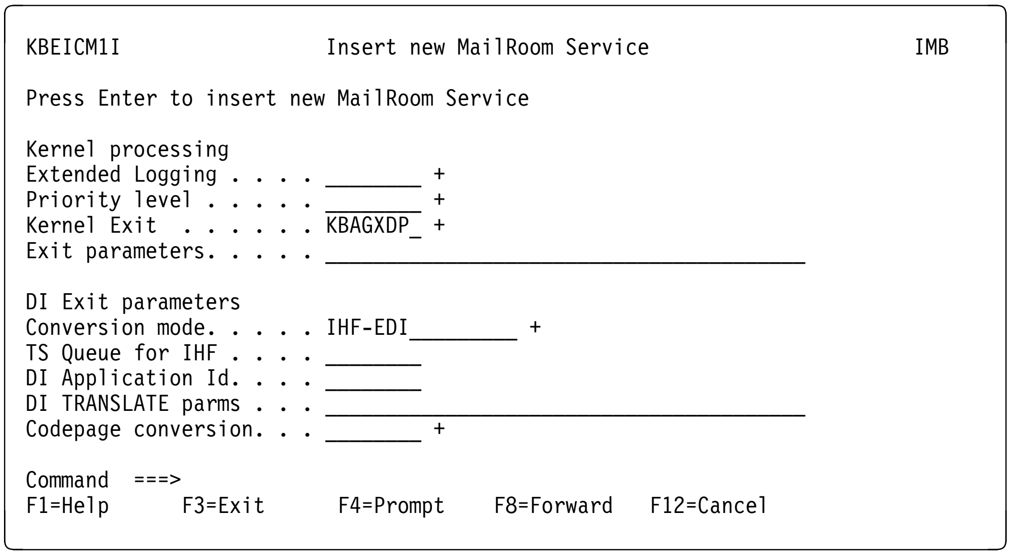
Figure 8. Setting up the codepage conversion exit parameters

You must specify the following value. You can get help the field by pressing F1.

Codepage conversion
: The name of a codepage conversion table. The name `CPee1ee2` identifies the origin codepage (ee1) and the target codepage (ee2). Both codepages will normally be EBCDIC codepages.

DataInterchange translation exit KBAGXDP

Kernel or destination exit used to let DataInterchange perform translation of a document. The example shown here uses the kernel exit. Use the Insert New MailRoom Service panel to set up the DataInterchange exit parameters. Figure 9 shows the relevant section of the panel.



```
KBEICM1I                Insert new MailRoom Service                      IMB

Press Enter to insert new MailRoom Service

Kernel processing
Extended Logging . . . . ________ +
Priority level . . . . . ________ +
Kernel Exit  . . . . . . KBAGXDP_ +
Exit parameters. . . . . ________________________________________

DI Exit parameters
Conversion mode. . . . . IHF-EDI_________ +
TS Queue for IHF . . . . ________
DI Application Id. . . . ________
DI TRANSLATE parms . . . ________________________________________
Codepage conversion. . . ________ +

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F8=Forward   F12=Cancel
```

Figure 9. Setting up the DataInterchange exit parameters

You should consider specifying the following values. You can get help for each field by pressing F1.

Conversion mode

The Conversion mode defines how the Exit will translate the input file. Three modes are defined:

IHF-EDI	In-house Format to EDI format.
EDI-IHF	EDI data to in-house format.
MAIL-EDI-IHF	EDI data (in an e-mail) to in-house format.

The translated document is kept as an updated version of the original document.

TS Queue for IHF

This queue name must be the same name as described under usage in DataInterchange. The queue is where the in-house format data is placed by DataInterchange after translation.

DI Application Id

The DI Application Id is used by DataInterchange to select which log file to use.

DI TRANSLATE parms

The DI TRANSLATE parms are used to append commands to the "PERFORM TRANSLATE" command issued when the documents are translated by DataInterchange.

Codepage conversion

If the destination document must be delivered in a different codepage, it can be entered here. The codepage file in CICS must be loaded. The format is CPfffttt where fff is the from and ttt is the to codepage. A more detailed explantion is available under online help.

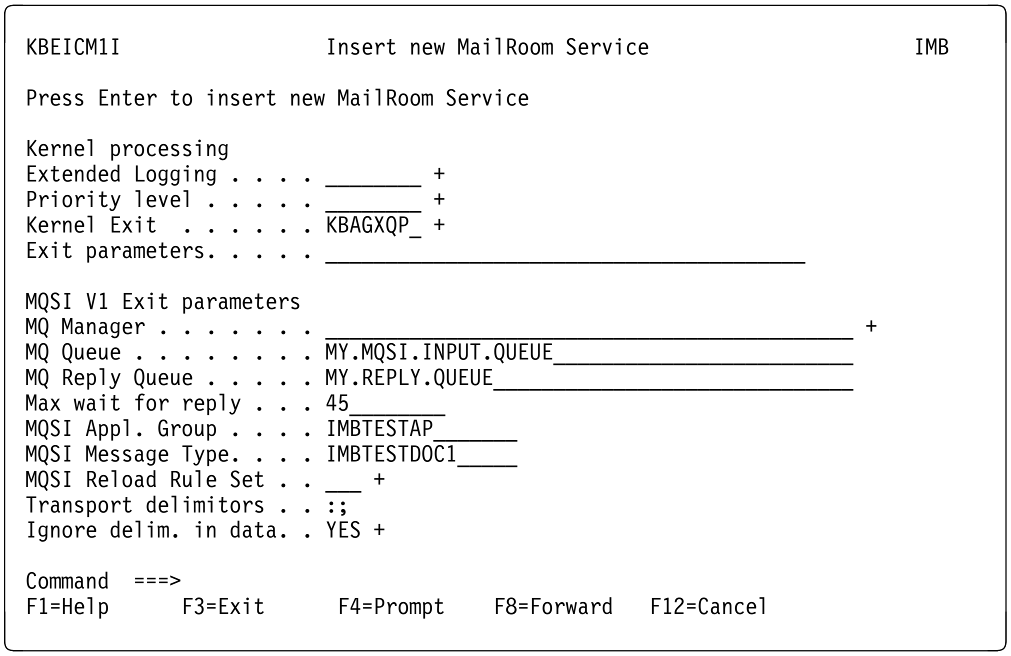
Usage notes

It is very important to specify the IHF TS queue name when translating to in-house format, as IMB otherwise can not find the translated data.

MQSI V1 exit KBAGXQP

Kernel or destination exit used to let an MQSI perform remapping of a document. MQSI should normally run in the same physical machine as IMB. The exit will format a document to a stream format (records separated with character delimitors), put it on a queue, wait for MQSI to reformat it, get the reformatted stream and finally return it to MailRoom as an updated version of the document.

The example shown here uses the kernel exit. Use the Insert New MailRoom Service panel to set up the MQSI exit parameters. Figure 10 on page 63 shows the relevant section of the panel.



```
KBEICM1I                 Insert new MailRoom Service                     IMB

Press Enter to insert new MailRoom Service

Kernel processing
Extended Logging . . . . ________ +
Priority level . . . . . ________ +
Kernel Exit  . . . . . . KBAGXQP_ +
Exit parameters. . . . . ________________________________________

MQSI V1 Exit parameters
MQ Manager . . . . . . . ____________________________________________ +
MQ Queue . . . . . . . . MY.MQSI.INPUT.QUEUE_________________________
MQ Reply Queue . . . . . MY.REPLY.QUEUE______________________________
Max wait for reply . . . 45________
MQSI Appl. Group . . . . IMBTESTAP_______
MQSI Message Type. . . . IMBTESTDOC1_____
MQSI Reload Rule Set . . ___ +
Transport delimitors . . :;
Ignore delim. in data. . YES +

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F8=Forward   F12=Cancel
```

Figure 10. Setting up the MQSI exit parameters

You should consider specifying the following values. You can get help for each field by pressing F1.

MQ Manager
Name of the MQ manager, normally blank for the local one

MQ Queue Name of the MQSI input queue. MQSI should automatically read the document from this queue.

MQ Reply Queue
Name of the queue that IMB should get the reply from. MQSI must be defined to put the reply to this queue. No triggering should be defined, this queue must not trigger the normal MQ source scenario.

Max wait for reply
The maximum time in seconds that the exit will wait for a reply from MQSI. If this amount of time has passed and the remapped reply has not arrived, the exit will assume that remapping failed and the MailRoom request will also be failed. In this case error determination must be performed in the MQSI environment.

MQSI Application Group
Name of the MQSI Application Group for this message.

MQSI Message Type
Name of the MQSI Message Type for this message.

MQSI Reload Rule Set
Should the exit ask MQSI to reload the ruleset. This setting will not reload the format.

Transport delimitors
Two characters used by the exit to build the buffer sent to MQSI and to decode the reply. First character (default **:**) will be placed between records in the sent buffer and is expected in the reply between

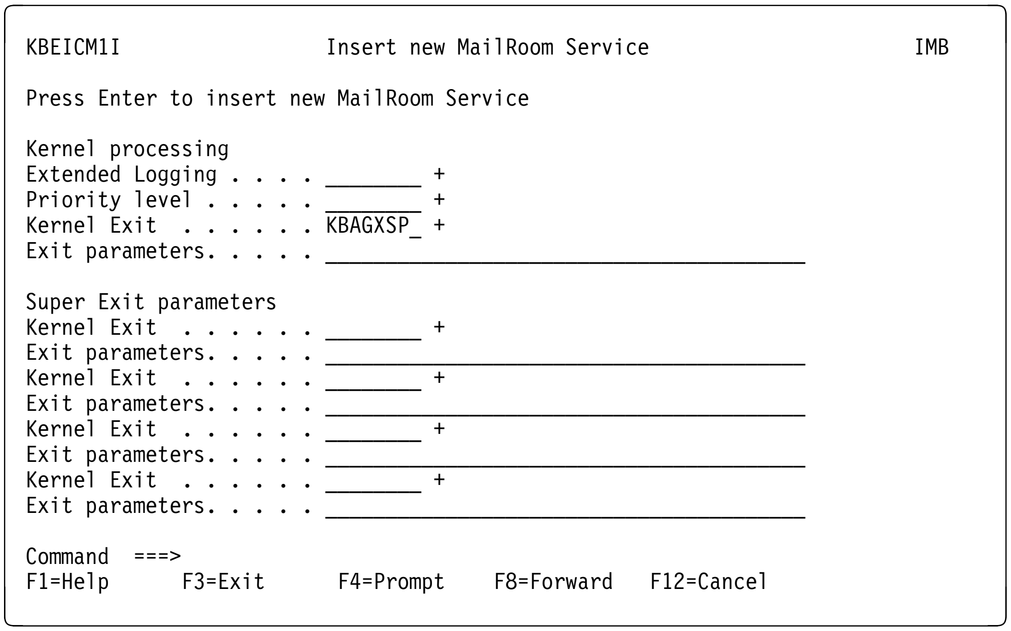
records. Second character (default **;**) will be placed at the end of the buffer being sent, it is not expected in the reply.

Ignore delimitors in data
: What should the exit do if the defined delimitors also occur in the input document. The exit can either ignore them (blank out) or fail the document.

Super exit KBAGXSP

Kernel or destination exit used to call a number of exits in sequence. The example shown here uses the kernel exit. Use the Insert New MailRoom Service panel to set up the Super exit parameters. Figure 11 shows the relevant section of the panel.

When using the Super exit up to five other exits can be specified. All exits have their own exit parameters. The exits can each return up to five event messages, but only the last five events returned to the super exit will be stored in MailRoom.



```
KBEICM1I                Insert new MailRoom Service                     IMB

Press Enter to insert new MailRoom Service

Kernel processing
Extended Logging . . . . ________ +
Priority level . . . . . ________ +
Kernel Exit  . . . . . . KBAGXSP_ +
Exit parameters. . . . . ________________________________________

Super Exit parameters
Kernel Exit  . . . . . . ________ +
Exit parameters. . . . . ________________________________________
Kernel Exit  . . . . . . ________ +
Exit parameters. . . . . ________________________________________
Kernel Exit  . . . . . . ________ +
Exit parameters. . . . . ________________________________________
Kernel Exit  . . . . . . ________ +
Exit parameters. . . . . ________________________________________

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F8=Forward   F12=Cancel
```

Figure 11. Setting up the Super exit parameters

XML Parsing to flat file exit KBAGXXP

Kernel or destination exit used to convert XML documents to a simple flat record oriented file. The exit is also defined as a display exit, where it will write semilar information to the panel in a more compact format.

The exit parses the XML document and write a D-record for every XML element (both beginning and end), XML attribute as well as character data. There is one record written for every event generated by the XML parser.

The following records/events are written:

```
Event             Record Type     Record layout
===================================================
Start of Document DOCSTART           0
PI Element        PIELEM             1
PI Attribute      PIATTR             3
Start of Element  ELEMSTART          1
Atribute          ATTRDATA           3
Characters        DATA               2
End of Element    ELEMEND            1
End of Document   DOCEND             0
```

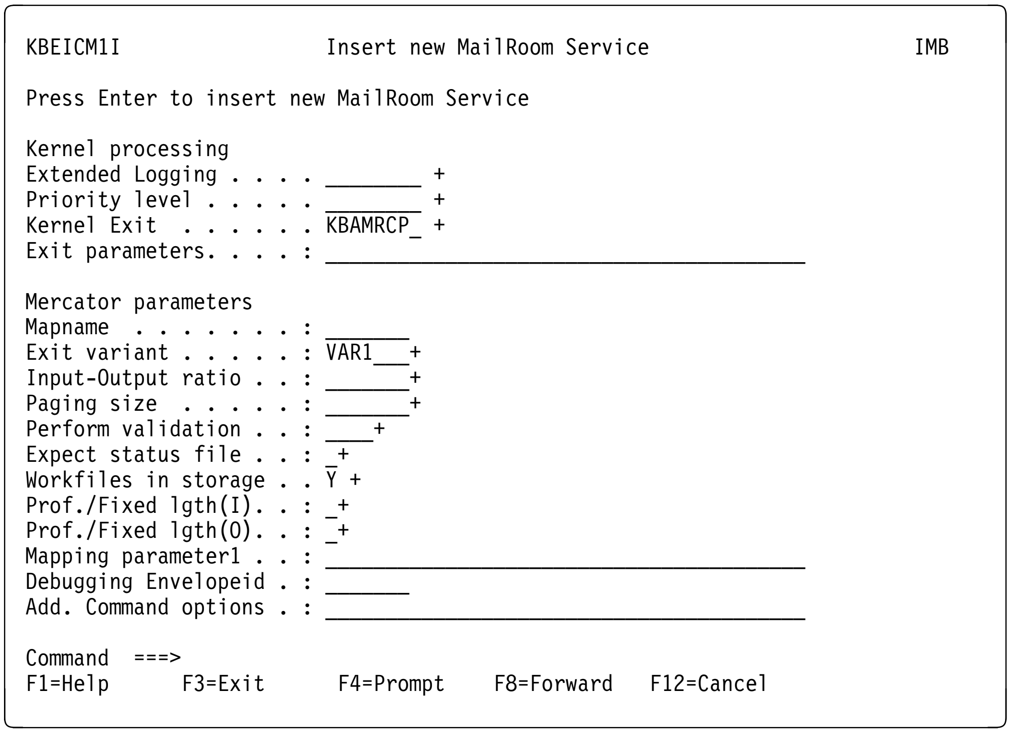
	D	Record Typ	Elem	Attr/Info	Length	Buffer	Path
0	D	Rec Ch(16)	reserved				
1	D	Rec Ch(16)	Elem Ch(20)	Info Ch(4)	reserved		Path Ch(200)
2	D	Rec Ch(16)	Elem Ch(20)	reserved	L Ch(5)	Buffer Ch(200)	Path Ch(200)
3	D	Rec Ch(16)	Elem Ch(20)	Attr Ch(20)	L Ch(5)	Buffer Ch(200)	Path Ch(200)

Description of fields:

Record Id	Character(1) Always D
Record Type	Character(16) Type of this record/event, record layout is dependent of this value.
Element name	Character(20) Name of current XML element
Info	Character(4) Contains value ROOT if current element is the root element
Attribute	Character(20) Name of attribute for current XML element
Length	Character(5) NNNNƀ 4 numbers (and a blank) to indicate length of data in buffer
Buffer	Character(200) Variable length buffer containing element data or attribute value. Longer values gets truncated.
Path	Character(200) The full path to current XML element showing the nesting levels.

Mercator exit KBAMRCP

Kernel or destination exit used to reformat data. The example shown here uses the kernel exit. Use the Insert New MailRoom Service panel to set up the Mercator exit parameters. Figure 12 on page 66 shows the relevant section of the panel.



```
KBEICM1I                 Insert new MailRoom Service                      IMB

Press Enter to insert new MailRoom Service

Kernel processing
Extended Logging . . . . ________ +
Priority level . . . . . ________ +
Kernel Exit  . . . . . . KBAMRCP_ +
Exit parameters. . . . : ________________________________________

Mercator parameters
Mapname  . . . . . . . : _______
Exit variant . . . . . : VAR1___+
Input-Output ratio . . : _______+
Paging size  . . . . . : _______+
Perform validation . . : ____+
Expect status file . . : _+
Workfiles in storage . . Y +
Prof./Fixed lgth(I). . : _+
Prof./Fixed lgth(O). . : _+
Mapping parameter1 . . : ________________________________________
Debugging Envelopeid . : _______
Add. Command options . : ________________________________________

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F8=Forward   F12=Cancel
```

Figure 12. Setting up the Mercator exit parameters

You should consider specifying the following values. You can get help for each field by pressing F1.

Extended Logging
The VERBOSE option inserts the Mercator Execution Log as event text in the MailRoom.

Exit variant The Exit variant defines how the Exit will set up the call to Mercator and how input files and output files will be processed. Four variants are defined:

Exit variant	**Description**
VAR1	I1:Input O1:Mapped doc.
VAR2	I1:Input O1:Mapped doc. O2:New document
VAR3	I1:Input O1:Mapped doc. O2:Error
VAR4	I1:Input O1:Mapped doc. O2:New doc. O3:Error

All variants have the MailRoom document as primary input and the mapped document as output. There can be two additional types of output defined and created by the Mercator map, *Error* and *New Document*.

Requesting the Mercator Exit to *expect status file* or using the *mapping parameter* will add an additional output card or input card.

The map variants must be set up in Mercator correspondingly.

Input-Output ratio
Used by the exit to calculate the expected size of the output. If the output is calculated to be larger than 32K records, the exit must use a VSAM data set for output.

Paging size
This parameter relates to performance.

Perform validation
Validation is performed on input data only. Validation of input data can be set on or off using this field.

Expect status file
Is a way for the Mercator exit to pass return codes and messages to the exit and to the MailRoom. The Status output card must be the last card. The Status file must consist of only two records:

1st record CHAR(3)
Mercator map return code.

2nd record CHAR(VAR)
Return message.

If the return code is not **000** the mapping is considered *FAIL*ed in the MailRoom. The return message is always shown as an event in the MailRoom.

Workfiles in storage
Is a way to ask Mercator to keep all workfiles in storage. The default is for work files to not be kept in storage. For small documents, keeping work files in storage will improve performance.

Prof./Fixed lgth(I)
This field is used to control how Mercator handles variable-length records on input. There are two possible values:

- P
- W

The value P causes the exit to add the extension ':P' to the input card. P is the default if the field is not filled. This extension means that the data will be 'profiled', ie a table will be built in memory which cross-references record identifiers to corresponding file offsets.

The value W indicates that the extension ':W' will be used, and Mercator will then copy the input to a fixed format workspace before mapping it. This may give better performance in some cases.

Prof./Fixed lgth(O)
This field is used to control how Mercator handles variable-length records on output. There are two possible values:

- P
- W

The value P causes the exit to add the extension ':P' to the output card. P is the default if the field is not filled. This extension means that the data will be 'profiled', ie a table will be built in memory which cross-references record identifiers to corresponding file offsets.

The value W indicates that the extension ':W' will be used, and Mercator will then map the data to a fixed format workspace. After mapping completes, variable length records are separated out and written to the TS queue. This may give better performance in some cases.

Mapping parameter1

This is a way to pass user-determined values to the Mercator map.

If this field is not blank, this field and all the attributes on the MailRoom service are passed to the map as a TS Queue on a second input card. The record layout of the TS Queue passed in the second input card is:

Name CHAR(16)
Contains field name of MailRoom Service.

Value CHAR(VAR)
Contains value of field.

This field can be filled for two reasons:

1. In order to make available to the Mercator map, all the attributes on the service and their values in the format described above. In this case the field can contain any non-blank character.
2. In order to pass a certain string or value to the Mercator map. In this case the map should read the service attribute called 'Mapping Parameter1' and process the contents. All the other service attributes will also be passed even if they are not required.

The map can then be set up to read, from the second input card, any value on the service, and/or the string entered by the user.

Debugging EnvelopeID

For Mercator debugging (execution, audit log, input trace, and output trace), you must restart the Envelope with the EnvelopeID in this field. The debugging output will be generated and sent to the Electronic Addresses defined as the MailRoom Service Point of Contact.

Add. Command options

This field allows the experienced Mercator user to enter a string which will be added to the Mercator call. The content of this field is a the user's risk and must be in the correct format including punctuation and spaces in order for Mercator to use it. In other words, no checking is performed on the string, but if filled it is appended to the Mercator call built up by the exit.

Usage notes

- All input and output records must be defined in the Mercator Type trees with terminator LF/CR=‘0D 0A’ Hex. Input and output records must be set to Hex in both *View as* and *Keep*.
- Error output will be typically be generated by the Mercator *RELEASE* setting and the *REJECT* function. Error output is supported in Mercator Exit variant *VAR3* and *VAR4*. If the Error output is *none empty*, the output will be sent to the Electronic Addresses defined as the MailRoom Service Point of Contact.
- A new document can be created by the Mercator map (not the mapped data) and will be processed by IMB in Mercator Exit variant *VAR2* and *VAR4*.

  Such a new document will be inserted into the MailRoom and sent to the destination defined in the leading M-record. The document must always start with an M-record and all following records must contain a *D* in the first position.
- Mercator MVS/CICS Version 1.4.2 or higher is needed for support of input larger than 32K records as well as for support of workfiles in storage.

  Refer to the *Installation Guide* for information about which version of Mercator to use.

Destination exit for OTMA support KBGXOTP

Destination exit used to send document to IMS using MQSeries and OTMA. The exit can only be used when destination is MQSeries.

The exit is used to build the MQIIH and IMS data header as the first part of the message data. The exit is using control information from the Dest Accounting Info field. This field must contain information about the IMS transaction, IMS user ID and RACF password.

Specification of control information in the Dest Accounting field is shown in Figure 13. The information is specified as part of the receive subscription.

```
Dest Accounting Info . : ttttttttuuuuuuuupppppppp
                         where tttttttt is IMS transaction
                               uuuuuuuu is IMS user ID
                               pppppppp is RACF password
```

Figure 13. MailRoom control information for OTMA exit.

Sample Display exit KBGXIDP

Exit used from MailRoom Document Versions panel to reformat contents of the document before display.

The exit will reformat some records in document such as the M-record (simple and extended), the SAP EDI_DC record and a number of DI records. For each of these records the exit will display the fields in the record one by one.

Record length Display exit KBGXIRP

Exit used from MailRoom Document Versions panel to show record length and total length of the document.

The exit will show the length of every record in the document and a total length of the whole document at the end.

Chapter 3. MailRoom CICS APIs

The IMB MailRoom can be accessed directly from a CICS program, by using the following CICS APIs. The APIs are CICS programs that can be called from any program running in CICS, which can EXEC CICS LINK to another program. All parameters are passed in the CICS communication area and the data is placed in CICS TS queues. For the Read API a CICS TD queue is also involved. This means that the APIs can be called from a remote CICS using DPL and remote definitions of CICS TS and TD queues.

The system acknowledgment scenario in CICS is handled by the read API.

Write API
Used to put documents into MailRoom

Read API
Used to read documents from MailRoom

Acknowledgment API
Used to give Business Acknowledgment to MailRoom

Document Browser API
Used to read documents in the MailRoom without performing any of the MailRoom processes.

CICS MailRoom write API—KBAXWRP

The write API can be used by business applications to have documents passed to the MailRoom without going through DataInterchange or Expedite/CICS.

The API can process single or multiple documents from an input TS queue, which can either be a single TS queue or an LTSQ.

It will trigger logging on the VSAM data set that can later be used for ASCA reconciliation.

Format

KBAXWRP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBAXWRP')
                    COMMAREA(KBAXWRIT)
                    LENGTH(CSTG(KBAXWRIT)); ──────────────────►◄
```

Parameters

Passed structure for the CICS MailRoom write API. It can be found in KBH.R450.PLINCL(KBAXWRIT)

```
/* +--------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------     | */
/* | (C) COPYRIGHT IBM DENMARK. 1998.      ALL RIGHTS RESERVED.   | */
/* | (C) COPYRIGHT IBM CORP. 1998.         ALL RIGHTS RESERVED.   | */
/* |                                                              | */
/* |                                                              | */
/* +--------------------------------------------------------------+ */

/*   +-----------------------------------------------------------+    */
/*   |              API STRUCTURE                                |    */
/*   |              =============                                |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBAXWRIT                                     |    */
/*   |                                                           |    */
/*   |  LENGTH    : 100 BYTES                                    |    */
/*   |                                                           |    */
/*   |  CONTENTS  : COMMUNICATION AREA TO CICS WRITE API KBAXWRP|    */
/*   |                                                           |    */
/*   |  RELATIONS : BUILT BY BUSINESS APPLICATIONS CALLING       |    */
/*   |              KBAXWRP                                      |    */
/*   |                                                           |    */
/*   +-----------------------------------------------------------+    */
/*                                                                    */
     5 SKBH_MSGID      CHAR(10), /* MESSAGE NUMBER                    */
     5 SKBH_TSQUEUE    CHAR(08), /* TS QUEUE NAME                     */
     5 SKBA_ENV_SRC    CHAR(12), /* MR ENVELOPE KEY - SOURCE          */
     5 SKBA_MULTI_TSQ  CHAR(03), /* LTSQ INDICATOR                    */
                                 /* IF 'YES' THE TSQ CONTAINS A LIST */
                                 /* OF TSQ NAMES INSTEAD OF DOCUMENTS*/
                                 /* EACH TSQ RECORD SHOULD THEN      */
                                 /* CONTAIN ONE NAME OF MAX 8 BYTES. */
     5 SKBH_COMMIT     CHAR(03), /* IF 'YES' THE WRITE API AUTOMATIC-*/
```

```
                                   /* LY, COMMIT/ROLLBACK AFTER INSERT */
                                   /* IN DB2 TABLES.                   */
     5 RESERVED        CHAR(64)  /* FOR FUTURE USE, BLANKS           */
 /*                                                                    */
 /*                                                 TOTAL LENGTH 100 */
 /*  ==  API  == END OF STRUCTURE KBAXWRIT ==                         */
 /*  ---------------------------------------------------------------- */
```

SKBH_MSGID
Message ID pointing to error message. If the message ID is blank, then the MailRoom write was successful and the application must commit the new data with a CICS SYNCPOINT. If the message ID is non blank, then the write was unsuccessful and the data must be rolled back with a CICS SYNCPOINT ROLLBACK.

SKBH_TSQUEUE
Name of TS-queue containing input documents to API. It can either be a LTSQ or a single TS queue. See also field: SKBA_MULTI_TSQ.

SKBA_ENV_SRC
Source envelope ID, returned by write API

SKBA_MULTI_TSQ
If this indicator is *YES*, the passed TS queue will be treated as a LTSQ (see: “Multiple TS queues” on page 21). Otherwise the TS queue will be treated as a single TS queue.

SKBH_COMMIT
If this indicator is *YES*, the API will commit the new data for every 1000 inserts with a CICS SYNCPOINT. This is only for inserts in the MailRoom TRANSPORT DOCUMENT DATA table. The final commit that makes data available for further processing must be done by the application as described in SKBH_MSGID.

RESERVED
Future use. Must be initialised to blanks by the calling application.

Old structure—KBAXWRW

This API can still be used without changing the application, and the API will not commit any new data. The previous API should not be used by any new application.

Format of input TS queue

- The data is passed to the API in a CICS TS queue.
- The queue can either be a LTSQ or a single TS queue. (See also: “Multiple TS queues” on page 21.)
- The input TS queue or LTSQ can contain multiple MailRoom documents that will be grouped together in an envelope.
- A MailRoom M-record, (either simple or extended) is used to indicate the beginning of each document. Other records must not have *M* in column 1, as the record then would be misinterpreted as an M-record.
- The very first record in the document must be an M-record.

It is the responsibility of the calling business application to use unique TS queue names that do not collide with other programs, and delete the input TS queue after the call to the API. When the LTSQ routines are used, the naming of TS queues is done automatically. For methods to manually generate unique TS-queue names, see “Generate unique TS queue names—KBHUQNP” on page 239.

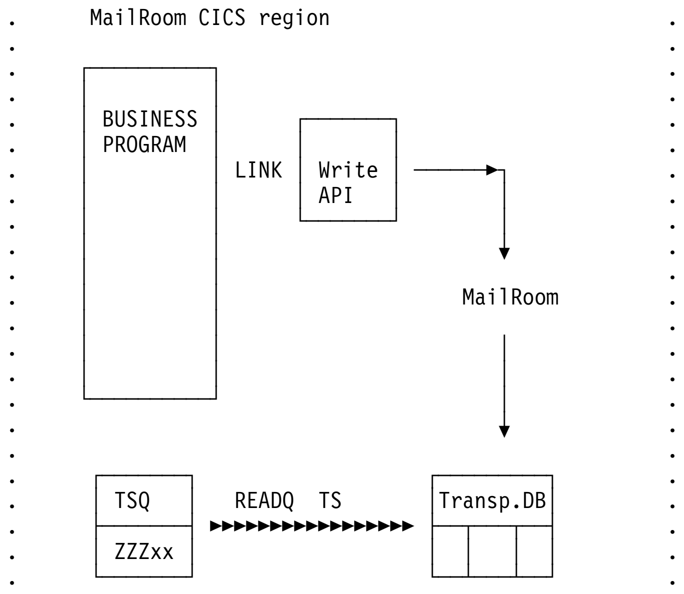
Examples

Examples of how to use the MailRoom write API are available in:

PLI	Sample using LTSQ: KBH.R450.PLI(KBGXW3M)
PLI	Sample using a single TS queue: KBH.R450.PLI(KBGXW4M)

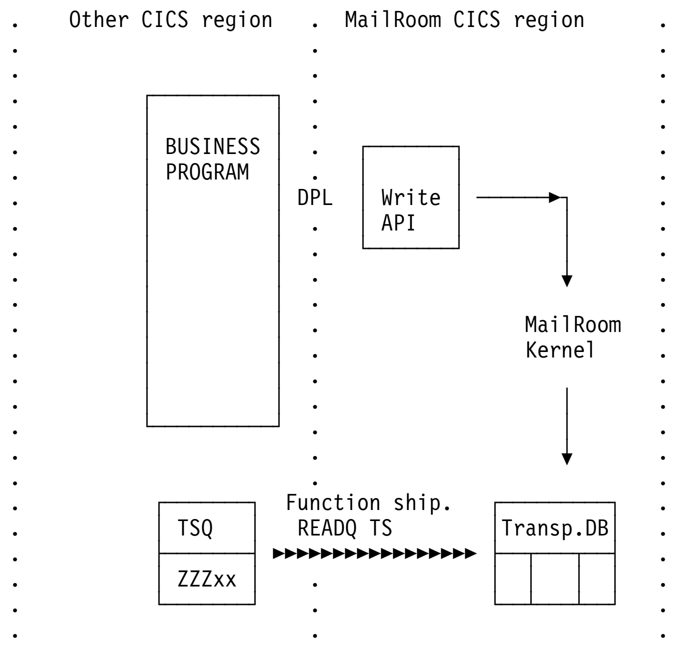
Business application and MailRoom in same CICS region

Usually the business application runs in the same CICS as the MailRoom and all resources are defined locally.



Business application and MailRoom in different CICS regions

Using CICS DPL (Distributed Program Link), a business application can run in another region and invoke the write API from there. In that case the TS-Queue should be written locally to a queue starting with ZZZ, and the MailRoom should be setup to read that particular TS Queue prefix ZZZ remotely (this is defined in the CICS Temporary Storage Table TST).



Messages

When system errors occur, for example, bind errors, or missing or disabled resources, the API will send a note to ISERROR.

Processing

The API will only perform a technical validation check (non-blank) of the M-record. A complete validation of the M-record data contents against MR registrations will be done later by the MailRoom Kernel and can then potentially result in a FAILED request with the document stored in the Transport Table.

- The M-record for every document is checked to see that mandatory parameters are filled in. That is, country code, trading partner number and MR layout. If there is a blank value in any of these 3 parameters, the document (all documents) will be rejected and an error code is returned to calling program.
- Having passed the non-blank check, the documents are inserted into the Transport Tables. They are related by the source envelope ID.
- A source envelope is inserted into the event tables, number of documents are registered.
- API now inserts log-records into VSAM file KBAASCA, telling how many documents were passed to the MailRoom in this particular Source envelope.
- The API returns Message ID and the Source envelope ID to caller.

In case of errors caused by any resource problem in the CICS system, then all documents in the source envelope must be removed from DB2 with a rollback, see below. ASCA logging is done, and a note is sent to ISERROR. It is up to the calling program to handle the error, and call the API later.

Syncpointing

The API will issue COMMIT for every 1000 inserts in MailRoom transport data table, if it is specified by the calling application. This is done by performing a `EXEC CICS SYNCPOINT` or a `EXEC CICS SYNCPOINT ROLLBACK` in the API. It is the responsibility of the calling application to perform the final syncpointing in all cases. This must be done according to the returned message code. If the returned message code is blank (successful call), then the application must perform a `EXEC CICS SYNCPOINT` to commit the updates done in DB2. If the returned message code on non-blank, then the application must perform a `EXEC CICS SYNCPOINT ROLLBACK` to ensure that IMB MailRoom DB2 tables are not left in an incomplete state.

If the CICS MailRoom write API is linked locally, the following DBRMs must be included in the DB2 Plan used by the application transaction:

- KBASTTM
- KBHUIDM
- KBHWCBM

Note: This is also the case if the API is called from a BPI. In that case the common BPI DB2 Plan KBIA2AP must also contain these DBRMs.

CICS MailRoom read API—KBAXREP

When the MailRoom decides that an envelope is to be processed by a business application (based on scheduling parameters for example), it triggers the business application by writing a record with the envelope key to a TD queue. (The TD queue has to be defined with trigger level = 1, in to let the document be processed immediately.)

The API is then used by the business application to read documents from the MailRoom.

The business application can optionally return one or more business acknowledgments to MailRoom during the processing of the document (see: “CICS MailRoom acknowledgment API—KBAXACP” on page 84).

CICS MailRoom read API is also providing the API support for the System Acknowledgment scenario in CICS (see the *System Administration Guide*). Used in this way, a CICS program can receive control after certain MailRoom conditions (Finished, Failed or Overdue status) and receive a special control document.

Format

KBAXREP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBAXREP')
                    COMMAREA(KBAXDST)
                    LENGTH(CSTG(KBAXDST)); ──────────────────►◄
```

Parameters

Passed structure for the CICS MailRoom read API. It can be found in KBH.R450.PLINCL(KBAXDST)

```
/* +-------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------     | */
/* | (C) COPYRIGHT IBM DENMARK. 1998.      ALL RIGHTS RESERVED.  | */
/* | (C) COPYRIGHT IBM CORP. 1998.         ALL RIGHTS RESERVED.  | */
/* |                                                             | */
/* |                                                             | */
/* +-------------------------------------------------------------+ */

/*   +---------------------------------------------------------+    */
/*   |              IMB MAILROOM                               |    */
/*   |              ==============================             |    */
/*   |                                                         |    */
/*   |  STRUCTURE : KBAXDST                                    |    */
/*   |                                                         |    */
/*   |  LENGTH    : 00500 BYTES                                |    */
/*   |                                                         |    */
/*   |  CONTENTS  : MAILROOM READ API STRUCTURE                |    */
/*   |                                                         |    */
/*   |  RELATIONS : READ API AND DESTINATION TD QUEUES         |    */
/*   |                                                         |    */
/*   +---------------------------------------------------------+    */
/*                                                                  */
/*                                                          OFFSET */
/*                                                             --- */
  5   KBAXDST_REC,                      /*                         */
   10  KBAXDST_RES1          CHAR(004), /* RESERVED             000 */
   10  SKBH_FNC              CHAR(008), /* FUNCTION             004 */
   10  SKBH_MSGID            CHAR(010), /* RETURN MESSAGE       012 */
   10  SKBH_TSQUEUE          CHAR(008), /* TS QUEUE             022 */
   10  SKBA_MASK_FNC         CHAR(008), /* DATA MASK FUNCTION   030 */
   10  SKBA_REQ_KEY,                    /* REQ KEY (FOR ACK)        */
    20  SKBA_ENV_DER         CHAR(012), /* DERIVED ENV KEY      038 */
    20  SKBA_DOCSEQNO_NUM    CHAR(010), /* DOC NUMBER           050 */
    20  KBAXDST_RES2         CHAR(018), /* RESERVED             060 */
   10  SKBA_ENV_SRC          CHAR(012), /* SOURCE ENV KEY       078 */
   10  SKBA_ENV_DST          CHAR(012), /* DEST ENVELOPE        090 */
   10  IOPUCTY               CHAR(003), /* OCUNTRY CODE         102 */
   10  ISYSIDY               CHAR(004), /* APPLICATION ID       105 */
   10  IPRAIDY               CHAR(008), /* SERVICE NAME         109 */
   10  ICUSPRM_FROM          CHAR(009), /* FROM TP NUMBER       117 */
   10  SKBA_LAYOUT           CHAR(016), /* LAYOUT OF DATA       126 */
   10  SKBA_TYPE_SRC         CHAR(008), /* SOURCE SCEN. TYPE    142 */
   10  SKBA_TYPE_DST         CHAR(008), /* DESTINATION SCEN. TYP 150 */
   10  SKBA_TOTLIN_NUM       CHAR(010), /* TOTAL NO OF LINES    158 */
   10  SKBA_MAXLNG_NUM       CHAR(010), /* MAX WIDTH OF RECORDS  168 */
   10  SKBA_ACKLVL           CHAR(001), /* ACKNOWLEDGMENT LVL   178 */
   10  SKBA_DST_DATA         CHAR(040), /* DESTINATION DATA     179 */
   10  ICUSPRM_TO            CHAR(009), /* TO TP NUMBER         219 */
   10  SKBA_M_REC_FORMAT     CHAR(008), /* FORMAT OF M-RECORD   228 */
   10  SKBA_MULTI_TSQ        CHAR(003), /* LTSQ INDICATOR       236 */
                                        /* USE 'YES' TO WORK WITH   */
                                        /* LARGE DOCUMENTS          */
   10  KBAXDST_RES3          CHAR(261)  /* RESERVED             239 */
/*                                                                 */
/*                                               TOTAL LENGTH 00500 */
/*  ==  END OF STRUCTURE KBAXDST  ==                               */
/*  ------------------------------------------------------------- */
```

KBAXDST_RES1
Reserved, should be initialized to blanks.

SKBH_FNC
The name of the function to be executed. Valid input values are:

- INIT - Initialization. Use this for the first invocation to read the structure. (Or read it directly from the TD queue.)
- READ - Read document into TS queue(s). MailRoom will write it into either a LTSQ or a single TS queue (depending on field SKBA_MULTI_TSQ).

SKBH_MSGID
A status field with the following mask XXXXXXYYY where XXXXXX is the program name, and YYY is the message number. (If OK - then blank)

SKBH_TSQUEUE
This is the name of the TS queue for the document used on the READ call to the API. Depending on the field SKBA_MULTI_TSQ it is used in the following way:

LTSQ - On input SKBH_TSQUEUE must contain a 4 character prefix, which will be used for all TS queues created. On return it contains the name of the LTSQ.

Single TS queue - On input SKBH_TSQUEUE must contain the name of a unique TS queue, where the document should be written. The same name is returned. The call will fail, if the size of the document exceeds one TS queue.

All TS-queue names beginning with 'KB' are IMB reserved names.

SKBA_MASK_FNC
In the READ-call, the calling business application can specify what types of data records are to be extracted from the transport tables. Valid types are:

MD	Only the MailRoom M-record and D-records will be returned. All other records (like C, E, I, Q records from DI) will be removed. This is default if SKBA_MASK_FNC is left blank.
RAW	All records saved with the document will be returned.
CD	A DI C-record and D-records will be returned. (If this option is used on transactions where a DI C-record is not available, then a DI compatible C-record will be created out of MailRoom control information.)
STRIP-MD	Only D-records will be returned and the D is stripped off. (This option can be used to retrieve a SAP IDOC, that always is stored with an M record and D in front of every line.)

SKBA_REQ_KEY
The complete transport table key for the document. It consists of the derived envelope key + the sequence number of the document within the envelope as described below.
Returned from TD queue or after INIT call.

SKBA_ENV_DER
The derived envelope key - part of Transport Table Key.
Returned from TD queue or after INIT call.

SKBA_DOCSEQNO_NUM
The document sequence number - part of Transport Table Key.
Returned from TD queue or after INIT call.

SKBA_ENV_SRC
The source envelope key.
Returned from TD queue or after INIT call.

SKBA_ENV_DST
The destination envelope key.
Returned from TD queue or after INIT call.

IOPUCTY
Country code.
Returned from TD queue or after INIT call.

ISYSIDY
Application system identifier.
Returned from TD queue or after INIT call.

IPRAIDY
MailRoom service identifier.
Returned from TD queue or after INIT call.

ICUSPRM_FROM
TP Account number (sender).
Returned from TD queue or after INIT call.

SKBA_LAYOUT
MailRoom document layout.
Returned from TD queue or after INIT call.

SKBA_TYPE_SRC
The source type of this envelope.
Returned from TD queue or after INIT call.

SKBA_TYPE_DST
The destination type.
Returned from TD queue or after INIT call.

SKBA_TOTLIN_NUM
This is the total number of records in the document. (This number can be larger than the actually number of records in the returned TS Queue depending on SKBA_MASK_FNC).
Returned from TD queue or after INIT call.

SKBA_MAXLNG_NUM
This is the largest length of any data record in the document.
Returned from TD queue or after INIT call.

SKBA_ACKLVL
The acknowledgment level specified on the service.
Returned from TD queue or after INIT call.

SKBA_DST_DATA
This is a 40 byte user field as specified on the service/subscription.
Returned from TD queue or after INIT call.

ICUSPRM_TO
TP Account number (receiver).
Returned from TD queue or after INIT call.

SKBA_M_REC_FORMAT
Format of the returned M-record must be specified on the READ-call.
Valid types are:

(blank)	The simple M-record is returned in the TS-queue.
EXTREC00	The extended M-record with both sender and receiver information is returned in the TS-queue.

SKBA_MULTI_TSQ
Format of the returned TS queue(s) must be specified on the READ-call.
Valid types are:

YES	User wants a LTSQ. Prefix is passed, full name is returned.
(other)	User wants single TS queue. Full name is passed and returned.

KBAXDST_RES3
Reserved for future use.

Read API usage by an IMB (local) application

The API is called twice:

- First with function = INIT, and the document-key is returned. An alternative method is to read the TD Queue directly.
- Next with function = READ. The document is returned in a TS queue. The calling program must select which type of TS queue to receive: LTSQ or single TS queue, and either pass prefix or a full unique TS queue name in field SKBH_TSQUEUE.
  More information about LTSQs can be found in “Multiple TS queues” on page 21.
  For methods to generate unique TS-queue names, see “Generate unique TS queue names—KBHUQNP” on page 239. The generation of queue names for a LTSQ are done automatically based on the prefix.

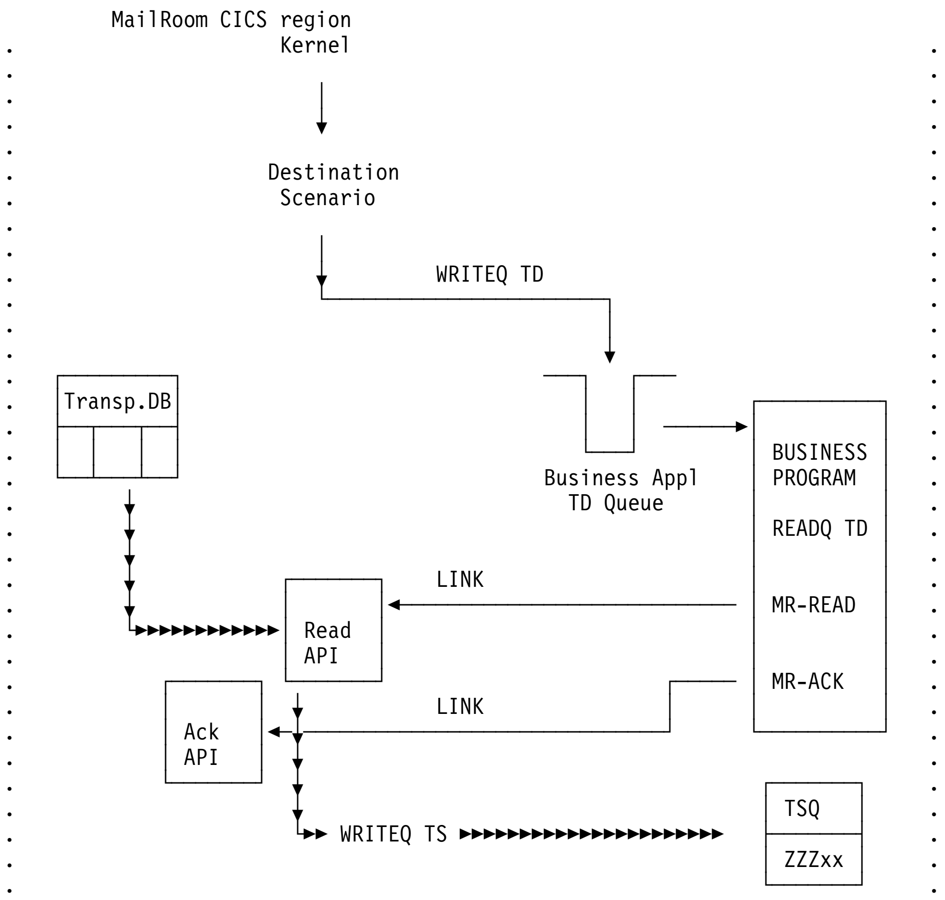
When the business application is through initialization and setup, it calls the read API using function INIT. The API reads the TD queue and then returns the KBAXDST structure updated with the full document key.

The business application should then

- Change the function verb to READ
- Specify TS queue type and name (prefix or full name)
- Specify the MASK function
- Specify the format of the M-record, and
- Call the API again
- Optionally return one or more business acknowledgments

The MailRoom will write the records pertaining to this document in the named TS queue or LTSQ and return control.

The business application then processes the data in the single TS queue or LTSQ.

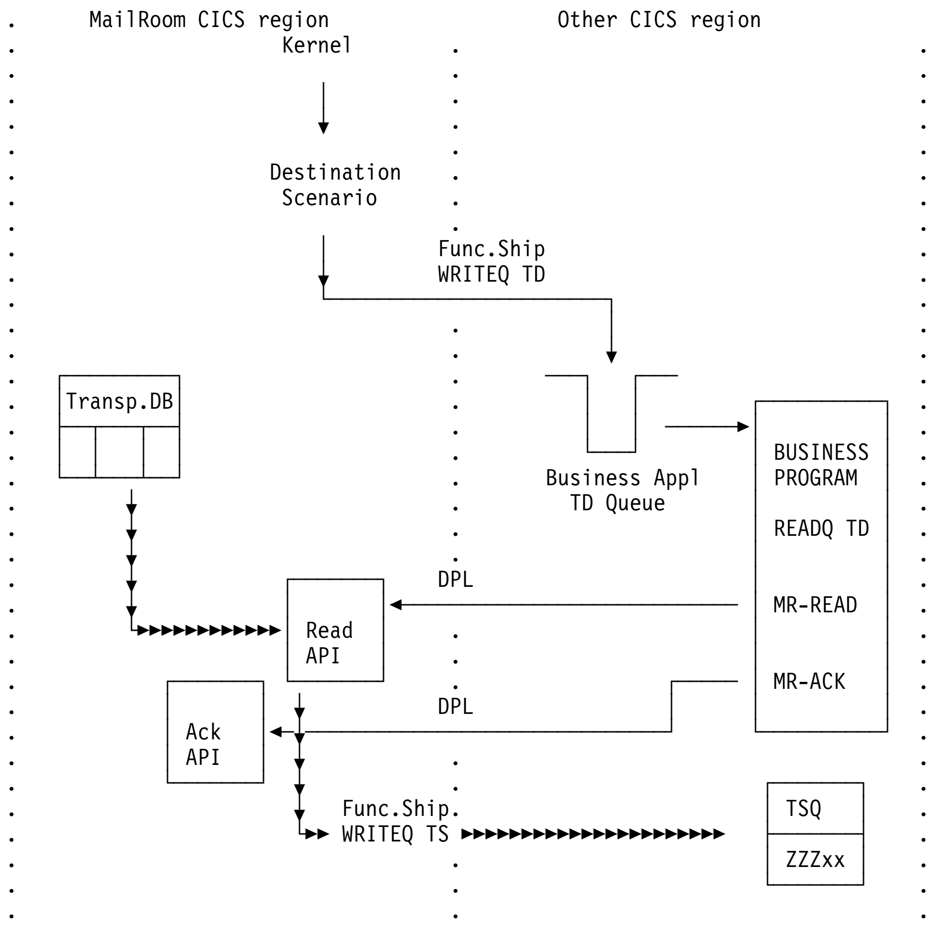


Read API usage by an IMB (remote) application

The business application that is to receive the MailRoom documents can also execute in an other (remote) region. In this case the triggering TD Queue (specified during Service registration) is defined in the MailRoom CICS as remote. When the business application in the remote region is triggered, it should not use the INIT-verb.

Instead it will read the TD Queue itself, using standard CICS READQ TD, into the (same) KBAXDST structure, initialize the parameters for the read API, and finally DPL Link to the KBAXREP (the read API itself has to execute in the MailRoom region.) The TS queues should physically be located in the remote CICS.

The API is only called once, with function = READ



CICS System Acknowledgment Scenario

When using the read API in a CICS System Acknowledgment Scenario, the received document will only contain one record, which is described in the *System Administration Guide*, and can be found in KBH.R450.PLINCL(KBAXSYSA). All the same rules apply in this case, except that only single TS queues are supported and business acknowledgments can not be made.

Examples

Examples of how to use the MailRoom read API are available in:

PLI	Sample read program using LTSQ and Ack API (Ack lvl 2): KBH.R450.PLI(KBGXR5M)
PLI	Sample read program using single TS queue (Ack lvl 0 or 1): KBH.R450.PLI(KBGXR7M)
PLI	Sample Sysack program using single TS queue and LTSQ: KBH.R450.PLI(KBGXSYM)

If the CICS MailRoom read API is Linked locally, the following DBRMs must be included in the DB2 Plan used by the application transaction:

- KBADTTM
- KBADSSM

CICS MailRoom acknowledgment API—KBAXACP

The API can be used by a server application to send back an acknowledgment to the MailRoom if the service/subscription was using acknowledgment level 2.

The application can send one or many acknowledgments back per MailRoom transaction, to record the progress of the particular transaction through the business process. Every acknowledgment will be stored as an event in the MailRoom Status tables. Sending many events back might introduce unnecessary processing overheads.

The API described here can either be used directly from CICS, or the specified structure can be sent from remote systems (MQSeries, TCP/IP, TIE and APPC). See:

- “Sending a Business Acknowledgment to MailRoom using MQSeries” on page 107
- “Sending a Business Acknowledgment to MailRoom using TCP/IP” on page 130
- “Sending a Business Acknowledgment to MailRoom using APPC” on page 138
- “Sending a Business Acknowledgment to MailRoom using TIE” on page 118

The data passed back to the MailRoom consists of:

- Process Indicator
- Optional Message data
- Optional Application Reference data

Process Indicator

The process indicator can have three values: I, F, or X.

I results in the MailRoom status for this transaction being changed from Pending to In-Process.

The last (or one-and-only) acknowledgment must provide a process indicator of *F* to indicate Finished, or *X* to indicate a severe problem and fail the document.

Message data

The application can optionally specify a Message number (on the MailRoom Message DB) and up to three variables to be put into the message. The resulting message will be used as the event text on the MailRoom status table, and (in case of failed) as text on the alert note.

Application Reference Data

The application can optionally return some *key* data in a 40 character field. This could be for example, a resulting invoice number, a computed delivery date or an amount. This field is stored by MailRoom in the request table and can later be viewed using the request status panels in the MailRoom administration panel hierarchy.

MailRoom service registration dependency.
The issuance of acknowledgments to the MailRoom has to be a planned activity, coordinated between the MailRoom administrator and the application developer.

When the MailRoom administrator registers an acknowledgment level of 2 against a particular MailRoom service; the MailRoom is waiting for an acknowledgment, and will take the overdue action if an acknowledgment doesn't arrive in time.
On the other hand - if an acknowledgment is received by the MailRoom on a service where the administrator has chosen an acknowledgment level of 0 or 1; the MailRoom will likewise object and store an *Acknowledgment out of sequence* event in the Status table.

The only exception to this is the TIE scenarios, where the acknowledgement can be returned from TIE to satisfy an acknowledgment level of 1.

Format

KBAXACP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBAXACP')
                    COMMAREA(KBAXACK)
                    LENGTH(CSTG(KBAXACK)); ──────────────────►◄
```

Parameters

Here is the passed structure for the CICS MailRoom Acknowledgment API. It can be found in KBH.R450.PLINCL(KBAXACK)

```
/* +-------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------    | */
/* | (C) Copyright IBM Denmark. 1998.      All Rights Reserved.  | */
/* | (C) Copyright IBM Corp. 1998.         All Rights Reserved.  | */
/* |                                                             | */
/* |                                                             | */
/* +-------------------------------------------------------------+ */

/*   +----------------------------------------------------------+    */
/*   |              IMB MAILROOM                                |    */
/*   |              ==============================              |    */
/*   |                                                          |    */
/*   |  STRUCTURE : KBAXACK                                     |    */
/*   |                                                          |    */
/*   |  LENGTH    : 00400 BYTES                                 |    */
/*   |                                                          |    */
/*   |  CONTENTS  : MAILROOM ACKNOWLEDGMENT API STRUCTURE       |    */
/*   |                                                          |    */
/*   |  RELATIONS : ACK API (AND ACK TD QUEUE KBA0)             |    */
/*   |                                                          |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
  5   KBAXACK_REC,                    /*                             */
   10  KBAXACK_RES1         CHAR(004), /* RESERVED              000 */
   10  SKBA_REQ_KEY,                   /* REQ KEY (FOR ACK)          */
```

```
     20  SKBA_ENV_DER         CHAR(012), /* DERIVED ENV KEY       004 */
     20  SKBA_DOCSEQNO_NUM    CHAR(010), /* DOC NUMBER            016 */
     20  KBAXACK_RES2         CHAR(018), /* RESERVED              026 */
    10  SKBA_PROCESS_IND      CHAR(001), /* PROCESS INDICATOR     044 */
    10  SKBH_MSGID            CHAR(010), /* INPUT MESSAGE ID      045 */
    10  SKBH_MSGVAR,                     /* MSG VARIABLE              */
     20  SKBH_MSGVAR_X(3)     CHAR(025), /* 1, 2, 3               055 */
    10  SKBA_REF_DATA2        CHAR(040), /* REFERENCE DATA (UPD)  130 */
    10  SKBA_ACK_TMSTAMP      CHAR(026), /* RESERVED (API TMSTAMP)170 */
    10  KBAXACK_RES3          CHAR(204)  /* RESERVED              196 */
/*                                                                    */
/*                                                TOTAL LENGTH 00400 */
/*  ==  END OF STRUCTURE KBAXACK  ==                                  */
/*  ---------------------------------------------------------------- */
```

KBAXDST_RES1
Reserved, should be initialized to blanks.

SKBA_REQ_KEY
Complete MailRoom key as returned by the MailRoom Read API.

SKBA_PROCESS_IND
Process indicator: I, X or F. If none of the above, then the status is changed to 'IN PROC', but an error note is sent to ISERROR.

SKBH_MSGID
Message ID. If blank, and SKBH_MSGVAR is blank, then the MailRoom will use KBAACA011, -012 and -013 respectively, depending on the process indicator.

If SKBH_MSGID is blank but MSGVAR is not, then KBAACA010 is used together with the passed message varable (-s).

```
KBAACA010   UK  I  &1&2&3
KBAACA011   UK  I  Document in process by application
KBAACA012   UK  I  Document finished by application
KBAACA013   UK  A  Document failed by application
```

SKBH_MSGVAR
Three variables, each 25 character, that can be used to compose an application and occurrence specific message.

SKBA_REF_DATA2
Application Reference data. If non-blank value is used, MailRoom will store the contents in the request table. (There also exists an SKBA_REF_DATA field, optionally containing application data passed on from the client environment, originally invoking the MailRoom.

SKBA_ACK_TMSTAMP
Reserved, must be initialized to blank. The timestamp is initialized by the API and will contain the current time. The inserted event in IMB MailRoom status tables will display this time.

h5.KBAXACK_RES3
Reserved, must be initialized to blank.

CICS MailRoom Document Browser API—KBAXDBP

This API can be used by a business application that needs to read an envelope, of which it knows the key, without triggering any of the automatic MailRoom processes associated with the CICS read API. Additionally, it provides the ability to read a document which is already in FINISH status.

The API is called by the business application from a MailRoom exit or batch destination program, to read documents from the MailRoom.

CICS MailRoom Document Browser API can also be used in the System Acknowledgment scenario in CICS (see the *System Administration Guide*). Used in this way, a CICS program can receive control after certain MailRoom conditions (Finished, Failed or Overdue status) and use the Document Browser API to retrieve all or part of the document.

The document browser API only supports the use of LTSQs.

Format

KBAXDBP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBAXDBP')
                    COMMAREA(KBAXDBT)
                    LENGTH(CSTG(KBAXDBT)); ────────────────────►◄
```

Parameters

Passed structure for the CICS MailRoom Document Browser API. It can be found in KBH.R450.PLINCL(KBAXDBT)

```
/* +----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
/* | (C) COPYRIGHT IBM DENMARK. 1999.      ALL RIGHTS RESERVED.     | */
/* | (C) COPYRIGHT IBM CORP. 1999.         ALL RIGHTS RESERVED.     | */
/* |                                                                | */
/* |                                                                | */
/* +----------------------------------------------------------------+ */

/*   +----------------------------------------------------------+    */
/*   |             IMB MAILROOM                                  |    */
/*   |             ==============================                |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBAXDBT                                      |    */
/*   |                                                           |    */
/*   |  LENGTH    : 00300 BYTES                                  |    */
/*   |                                                           |    */
/*   |  CONTENTS  : MAILROOM DOCUMENT BROWSER API STRUCTURE      |    */
/*   |                                                           |    */
/*   |  RELATIONS :                                              |    */
/*   |                                                           |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
```

```
   5   KBAXDBT_REC,                   /*                          */
    10  KBAXDBT_RES1         CHAR(004), /* RESERVED            000 */
    10  SKBH_FNC             CHAR(008), /* FUNCTION            004 */
    10  SKBH_MSGID           CHAR(010), /* RETURN MESSAGE      012 */
    10  SKBH_LTSQ            CHAR(008), /* TS QUEUE            022 */
    10  SKBA_MASK_FNC        CHAR(008), /* DATA MASK FUNCTION  030 */
    10  SKBA_REQ_KEY,                   /* REQ KEY (FOR ACK)       */
     20  SKBA_ENV_DER        CHAR(012), /* DERIVED ENV KEY     038 */
     20  SKBA_DOCSEQNO_NUM   CHAR(010), /* DOC NUMBER          050 */
     20  KBAXDBT_RES2        CHAR(018), /* RESERVED            060 */
    10  SKBA_M_REC_FORMAT    CHAR(008), /* FORMAT OF M-RECORD  078 */
    10  KBAXDBT_RES3         CHAR(214)  /* RESERVED            086 */
 /*                                                             */
 /*                                         TOTAL LENGTH 00300 */
 /*  ==  END OF STRUCTURE KBAXDBT  ==                           */
 /*  ---------------------------------------------------------- */
```

KBAXDBT_RES1 Reserved, should be initialized to blanks.

SKBH_FNC Currently not used, should be initialized to blanks.

SKBH_MSGID A status field with the following mask XXXXXXYYY where XXXXXX is the program name, and YYY is the message number.
(If OK - then blank)

SKBH_LTSQ This is the name of the TS queue for the document. On input SKBH_LTSQ must contain a 4 character prefix, which will be used for all TS queues created. On return it contains the name of the LTSQ.

All TS-queue names beginning with *KB* are IMB reserved names.

SKBA_MASK_FNC Here the calling business application can specify what types of data records are to be extracted from the transport tables.
Valid types are:

MD Only the MailRoom M-record and D-records will be returned. All other records (like C, E, I, Q records from DI) will be removed. This is default if SKBA_MASK_FNC is left blank.

RAW All records saved with the document will be returned.

CD A DI C-record and D-records will be returned.
(If this option is used on transactions where a DI C-record is not available, then a DI compatible C-record will be created out of MailRoom control information.)

STRIP-MD
Only D-records will be returned and the D is stripped off.
(This option can be used to retrieve a SAP IDOC, that is always stored with an M-record and D in front of every line.)

SKBA_REQ_KEY The complete transport table key for the document. It consists of the derived envelope key + the sequence number of the document within the envelope.

The business application must fill this key in order to retrieve the document.

SKBA_ENV_DER The derived envelope key - part of Transport Table Key.

SKBA_DOCSEQNO_NUM The document sequence number - part of Transport Table Key.

SKBA_M_REC_FORMAT Format of the returned M-record must be specified.

Valid types are:

(blank) The simple M-record is returned in the TS-queue.

EXTREC00
The extended M-record with both sender and receiver information is returned in the TS-queue.

KBAXDBT_RES3 Reserved for future use, should be initialized to blanks.

Document Browser API usage by an IMB (local) appl

The calling program calls the API once, passing the key of the document required, the format of the M-record, the records required and the prefix of the LTSQ.

More information about LTSQs can be found in “Multiple TS queues” on page 21.

The generation of queue names for an LTSQ is done automatically based on the prefix.

CICS System Acknowledgment Scenario

If the business application uses the CICS Read API in the CICS System Acknowledgment Scenario, the received document will only contain one record, which is described in the System Administration Guide, and can be found in KBH.R450.PLINCL(KBAXSYSA). If the business application requires more of the document, it can then call the Document Browser API, using the key retrieved for the Read API call.

Examples

Examples of how to use the MailRoom read API are available in:

PLI Sample Sysack program using single TS queue and LTSQ: KBH.R450.PLI(KBGXSYM)

If the CICS MailRoom read API is Linked locally, the following DBRM must be included in the DB2 Plan used by the application transaction:

- KBADTTM

Chapter 4. MailRoom Batch utility programs

The IMB MailRoom can be accessed from a batch job using these utility programs.

Write Utility
: Used to put documents from a flat file into MailRoom.

Read Utility
: Used to read documents from MailRoom and put them into a flat file.

Write Utility using MQSeries.
: Used to format documents from a flat file in IMB MQSeries format and put them to an MQSeries queue which can be set up to trigger the MailRoom. This utility allows for simpler transmission of documents from a remote MVS system.

Read Utility using MQSeries.
: Used to read messages in IMB MQSeries format from a MQSeries queue and split the message into into records and write them into a flat file. This utility allows application on a remote MVS system to receive documents from IMB in a simple way using MQSeries.

Sample JCL for batch write utility—KBASBWX

The job is a Batch job, that can run under OPC control, and is therefore not under the control of the MailRoom. The job reads a flat file and inserts the documents into the MailRoom Transport Tables.

Format

KBASBWX is a JCL sample on how to use the utility. The job is in the RUN-library:

```
KBH.R450.RUN(KBASBWX)
```

The following DB2 plan is used by the Batch Write Utility:

- KBASBWP

Parameters

In the RUN member there is a short guidance on what to customise before running the job. Here is a description of the input to the utility:

Input variables to the Utility: The *INFILE* contains a debug parameter to the program followed by an originator field.

```
//INFILE   DD *
NO JOB=KBASBWX,CLASS=X,MSGCLASS=X,PROGRAM=KBASBWP,PLAN=KBASBWP
¬  ¬
|  |ORIGINATOR
|
|DEBUG
```

Debug The debug parameter is a 3 character long field, and is used to control whether you want to debug the Utility or not. If the debug parameter has the value 'YES', the Utility will write all log information to the SYSPRINT file.

Originator
The originator is a 80 character long field, and contains information about where the documents come from. The information is logged on the Source Envelope in the Originator field.

The sample job passes information about the original job, the program, the DB2 plan and the classes.

Any text can be passed as originator, as it is free format text.

Input

The *KBASBI* DD name contains the documents to the MailRoom. Here are some sample documents with simple M-records:

Sample 1—Same Trading Partner

```
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 1                        0000323
DKBAZNAD1       SU                  John
DKBAZNAD1       AnyCityAnyWhere     Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
```

```
DKBAZLIN7       000005
DKBAZTMA0       207002
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 2                      0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
```

The sample shows 2 documents to the same Country and Trading Partner using the same Service. The Utility will then tie the 2 documents together in the same Source Envelope, and same Derived Envelope.

Sample 2—Different Trading Partners

```
M123009999999KBAZBABA
DKBAZBGM0       384TestInvoice 1                      0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 2                      0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
```

This samples shows 2 documents with different Countries and Trading Partners, but with same layouts. The Utility will then split the 2 documents into 2 Source Envelopes containing one Derived Envelope each.

The Utility checks that Country code, Trading Partner and Layout from M-record are non-blank. If there is a syntax error or errors occur while executing, the program terminates and writes the error message to SYSPRINT file.

Sample JCL for batch read utility—KBADBRX

The job is an OPC job running during the night, and is therefore not under control of the MailRoom. The job reads an Envelope with a given search criteria from the MailRoom Transport Tables and writes it into a flat file.

Format

KBADBRX is a JCL sample on how to use the utility. The job is in the RUN-library:

```
KBH.R450.RUN(KBADBRX)
```

The following DB2 plan is used by the Batch Read Utility:

- KBADBRP

Parameters

In the RUN member there is a short guidance on what to customize before running the job. Here is a detailed description of the input to the utility:

Input to program

Input variables to the Utility: The *KBASYSIN* contains a debug parameter to the program followed by an originator field.

```
//KBASYSIN DD *
NO MD      EXTREC00REFDATA TO INPUT
¬  ¬       ¬       ¬
|  |       |       |
|  |       |       |-DESTINATION REFERENCE DATA TO BE PASSED
|  |       |         IN THE M-RECORD
|  |       |
|  |       |-EXTREC00 = EXTENDED M-REC, BLANK = SIMPLE M-REC
|  |
|  |-THE MASK TO THE READ API "RAW     "/"MD      "/"CD      "
|    RAW  - YOU WANT THE DATA AS IT IS IN THE TRANSPORT TABLE
|    MD   - YOU WANT THE M-RECORD AND THE DATA RECORDS
|           IT IS THE DEFAUL VALUE
|    CD   - YOU WANT THE READ API TO GENERATE THE C-RECORD
|           AND SEND IT TOGETHER WITH THE DATA RECORD
|
|-DO YOU WANT TO DEBUG THE PROGRAM "YES" OR "NO "
```

Debug The debug parameter is a 3 characters long field, and is used to control whether you want to debug the Utility or not. If the debug parameter contains the value *YES*, the Utility will write all log information in the SYSPRINT

Mask The Mask indicate how you prefer the data:

- RAW - All records saved with the document will be returned.
- MD - Only the MailRoom M-record and D-records will be returned. All other records (like C, E, I, Q records from DI) will be removed. This is default if MASK is left blank.
- CD - A DI C-record and D-records will be returned.

(If this option is used on transactions where a DI C-record is not available, then a DI compatible C-record will be created out of MailRoom control information.)

- STRIP-MD - Only D-records will be returned and the D is stripped off.
  (This option can be used to retrieve a raw SAP IDOC, that is always stored with an M-record and D in front of every line.)

Destination reference data
The destination reference data is written in the M-record and passed to the flat file.

Search criteria

The "KBADBI" file contains the search criteria to the Utility. The criteria is used to select which documents that are retrieved and written in the flat file.

```
APPLICATION='KBAZ',
COUNTRY='*  ',
TO_TRADING_PARTNER='*        ',
LAYOUT='*               ';
```

Application
The Application code field is mandatory and is not a generic search.

Country The Country code can be filled or it can be used with a generic search. When using generic search a "*" (asterisk sign) should be used.

TO_Trading_Partner
The Trading Partner code can be filled or it can be used with a generic search. When using generic search a "*" (asterisk sign) should be used.

Layout. The Layout code can be filled up or it can be used with a generic search. When using generic search a "*" (asterisk sign) should be used.

Output

The "KBADBO" will contain the documents written from the transport tables.

The Utility will make a crude check of the search criteria. If there is a syntax error or an error occurs during the execution, the program will stop executing and write the error message in the SYSPRINT.

Sample JCL for batch write via MQSeries utility—KBASMPX

This a Batch job, that can run under OPC control, and is therefore not under the control of the MailRoom. The job reads a flat file and builds a buffer in the standard MailRoom format. The buffer is put on an MQSeries queue and can from there be sent to the MailRoom. This job is can be used as an alternative to the Batch Write utility if the data to be sent to the MailRoom is on a different MVS system from the MailRoom.

Format

KBASMPX is a JCL sample on how to use the utility. The job is in the RUN-library:

```
KBH.R450.RUN(KBASMPX)
```

Parameters

In the RUN member there are instructions on what to customize before running the job. Here is a description of the input to the utility:

Input variables to the Utility: File *MQINFO* contains the parameters to the program.

```
//MQINFO   DD *
MQMANAGER=XXXX
MQQUEUE=QL.XXXXXX.XXXX.XX
DELIMITER=+++
RECFM=V
DEBUG=Y
SPLIT=Y
MATCH-REPLY=N
MRECORDS=Y
```

MQMANAGER
The name of the MQSeries queue manager on which the queue resides. This field is four characters as this is the maximum length of a queue manager name on MVS. The job must be run on the same MVS system as the MQSeries queue manager resides. This field is mandatory.

MQQUEUE
The name of the MQSeries queue on which the data is to be written. This field is 48 characters long. The queue must exist when the job runs. This field is mandatory.

DELIMITER
This field is three characters long and is used if the input file is F or FB. The string specified here is used to denote the end of a record. It is intended for use with input files created in an editor (for example ISPF) to ensure that the correct record length is passed to the program. The reasons for this are:

- Some recipients of documents from the MailRoom require records to keep to predefined lengths (for example ECMVS TIE)
- Excess space is trimmed away from the fixed length records before they are copied to the MQSeries message

Any value can be used, but you should use a string which never appears in your data. If the field is blank (or the delimiter string has not been added to the file) the records will all have the record length of the input file.

For file format V or VB this field is ignored.

This field is optional and defaults to blank if omitted.

RECFM This field is one character and must contain the character V to take advantage of the record length information if the input is V or VB. The field is optional but defaults to F if omitted.

DEBUG The debug parameter is a 1 character field, and is used to control whether debug information is printed. If the debug parameter has the value *Y*, the Utility will write all log information to the SYSPRINT file. Otherwise, only the summary report will be written to the SYSPRINT file. This field is optional, default N.

SPLIT The split parameter is used to control how documents for the same country are handled. If two documents in sequence belong to the same country number they will be placed in the same envelope unless this parameter is specified and set to Y. Y forces a new envelope each time a new M record is found. The field is optional and defaults to N.

MATCH-REPLY
The Match reply parameter is for use in situations where the message being put to MQSeries is a reply to another message, and there is a requirement to match the reply to the original message.

If this parameter is set to Y, the input file in DDname MQMDIN must contain the MQMD of the original message. The program will then use the Message Id of the original message to fill the Correlation Id of the new message, and the new message will be of type REPLY. As only one document can be used to reply to another, this option assumes that the input file contains only one document and any included M Records are not treated as the start of a new document. The SPLIT parameter is ignored if specified. M Records are not mandatory if this option is set to Y.

The field is optional and defaults to N. If Y is selected and DDname MQMDIN is missing or invalid, the program terminates without writing to MQSeries.

MRECORDS
This parameter allows the user to specify whether M Records are included in the data or not. If MRECORDS=N, no M Record checking is performed and the file is assumed to contain only one document. The SPLIT parameter will then be ignored if specified.

The field is optional and defaults to Y.

Input

The *MQINPUT* DD name contains the documents to write the MailRoom. The documents must have M-records and the data records must be prefixed by the MailRoom D prefix. The exception to the above is where the document is a reply to a previous MQSeries message and MATCH-REPLY is set to Y, or if MRECORDS=N. The maximum record length is 2000.

If the record format of the file is F or FB for ease of editing, a delimiter string as previously described is recommended. The string should be added after the last byte of each input record.

The *MQMDIN* DD name contains the MQMD of a previous MQSeries message to which this is the reply. This DDname is only required if parameter MATCH-REPLY=Y.

Here are some sample documents with simple M-records:

Sample 1—Same Trading Partner

```
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 1                       0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 2                       0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
```

The sample shows two documents to the same Country and Trading Partner using the same Service. The Utility will tie the two documents together in the same Source Envelope, and same Derived Envelope, unless the SPLIT parameter has been set to Y.

Sample 2—Different Trading Partners

```
M123009999999KBAZBABA
DKBAZBGM0       384TestInvoice 1                       0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
M678000800000KBAZBABA
DKBAZBGM0       384TestInvoice 2                       0000323
DKBAZNAD1       SU                    John
DKBAZNAD1       AnyCityAnyWhere       Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       000003
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
```

This sample shows two documents with different Countries and Trading Partners, but with same layouts. The Utility will split the two documents into two Source Envelopes containing one Derived Envelope each.

Sample 3—Fixed length input.

```
M123009999999KBAZBABA
DKBAZBGM0       384TestInvoice 1                      0000323
DKBAZNAD1       SU                   John
DKBAZNAD1       AnyCityAnyWhere      Travolta
DKBAZLIN7       000001
DKBAZLIN7       000002
DKBAZLIN7       THIS RECORD, SHORTER THAN THE REST, ENDS WITH A DASH-+++
DKBAZLIN7       000004
DKBAZLIN7       000005
DKBAZTMA0       207002
```

This sample shows how the delimiter +++ was used to indicate the end of a record which is shorter than the length of the file, where the file is fixed length. In this case the parameter DELIMITER=+++ is required.

If parameter MRECORDS is not N, and MATCH-REPLY is not Y, the Utility checks that Country code, Trading Partner and Layout from the M-record are non-blank. Errors in the M record result in return code 0004 and processing is stopped. Errors in the parameters or input files, or MQSeries errors, result in return code 0008. The job then terminates after writing the error message to SYSPRINT file. If no errors were found the job terminates with code 0000.

Large documents

The utility has been tested with buffer sizes of up to 100MB. This value is the current maximum message size that MQSeries can handle. Note that the actual number of bytes of application data is less than this as space must be allowed for the MQMD and the MailRooms own MQSeries header. When the program finds that it has to extend the buffer beyond 100MB, warning messages are issued to SYSPRINT. Note also that some installations of MQSeries can only handle max 4MB per message.

If large documents are processed regularly, the region size of the job must reflect this. In order to send the maximum buffer size, the largest region size available must be specified on the jobcard.

Note that if M-records are included, the SPLIT parameter allows for the file to be split into smaller documents, which may give better performance than one very large file. Alternatively, ensure that the input file is never larger than MQSeries can handle.

Sample JCL for batch MQSeries read utility—KBADMGX

The job is an OPC job running in batch, and is therefore not under control of the MailRoom. The job reads a message in MailRoom MQSeries format from a given queue and writes it into a flat file.

Format

KBADMGX is a JCL sample of how to use the utility. The job is in the RUN library:

```
KBH.R450.RUN(KBADMGX)
```

Parameters

In the RUN member there is a short guidance on what to customize before running the job. Here is a detailed description of the input to the utility:

Input variables to the Utility:

File *MQINFO* contains the parameters to the program.

```
//MQINFO   DD *
MQMANAGER=XXXX
MQQUEUE=XXXXXX.XXXX.XX
BUSACK=NO
MQQUEUE-ACK=XXXXXX.XXXX.XX.X
MASK-FNC=RAW
MATCH-REPLY=N
DEBUG=N
```

MQMANAGER
The name of the MQSeries queue manager on which the queue resides. This field is four characters as this is the maximum length of a queue manager name on MVS. The MQSeries queue manager must be on the same MSV system as the job runs. This field is mandatory.

MQQUEUE
The name of the MQSeries queue from which the data is to be read. This field is 48 characters long. The queue must exist when the job runs. This field is mandatory.

BUSACK This field is used to define whether the utility must issue a business acknowledgment upon reading the message successfully. The field is optional but defaults to N if omitted. If the field is present the parameter MQQUEUE-ACK is mandatory.

MQQUEUE-ACK
The name of the MQSeries queue to which the business acknowledgment is to be written. This field is 48 characters long. The queue must be defined. The field is only mandatory if the BUSACK parameter is set to YES.

MASK-FNC
This parameter governs which records will be written to the output file. An message read by this utility may contain various types of MailRoom record. The following list shows the possibilities for this field:

RAW All records will be written to the output file. This is the default value.

MD Only the M-record and all records beginning with D will be written.

CD Only the MailRoom C record and records beginning with D will be written.

STRIP-MD All records except the M-record will be written to the output file, and the 'D' will be stripped off data records.

CD-ALL All records except the M-record will be written to the output file.

Use of this parameter presupposes that the data arrives in MailRoom format. If the data to be processed does not contain normal MailRoom M and D records, it is important to set this parameter to RAW to avoid data records being interpreted incorrectly.

MATCH-REPLY
The Match reply parameter is for use in situations where the message being read from MQSeries requires a reply in the form of another MQSeries message, and there is a requirement to match the reply to the original message.

If this parameter is set to Y, the output DDname MQMDOUT must point to a dataset with format FB and record length 324. The program will then write the MQMD of the retrieved message to this file. To perform this function, the program can only handle one message (and therefore one MQMD) at a time and so, when this parameter is set to Y, only the first record will be read from the queue.

The field is optional and defaults to N. If Y is selected and DDname MQMDOUT is missing or invalid, the program terminates without reading from MQSeries.

DEBUG The debug parameter is a one character field, and is used to control whether debug information is printed. If the debug parameter has the value *Y*, the Utility will write all log information to the SYSPRINT file. Otherwise, only the summary report will be written to the SYSPRINT file. This field is optional, default N.

Output

The message(s) will be written to the DDname DOCOUT. This file must have record length 2000 and format V.

The Utility will make a crude check of the search criteria. If there is a syntax error or an error occurs during the execution (eg an &mqs. error) the program will stop executing and write the error message in the SYSPRINT.

0000 Data read from MQSeries queue and written to file.

0004 No messages on MQSeries queue.

0008 Error has occured. This can be syntax errors or errors during execution.

Chapter 5. MailRoom MQSeries support

The IMB MailRoom can be accessed using MQSeries. This chapter describes the IMB MailRoom support of MQSeries and also the required definition in MQSeries to access IMB MailRoom.

Send documents to MailRoom.
: How to use MQSeries to send documents to MailRoom.

Receive documents from MailRoom.
: How to use MQSeries to receive documents from MailRoom.

Send Business Acknowledgment to MailRoom
: How to use MQSeries to send Business Acknowledgment to MailRoom.

Data conversion
: MQSeries and data conversion in MailRoom.

MQSeries requirements
: Definitions in MQSeries to access the MailRoom.

MQSeries requirements for acknowledgment level 1 and 2
: Definitions in MQSeries to receive technical acknowledgement (delivery confirmation).

MQSeries sample code
: A sample program included in the IMB package showing examples of how to map application data to a buffer in the IMB MQSeries format.

Using MQSeries MQPUT to send documents to MailRoom

Using a standard MQPUT API call, an application on any supported MQSeries platform can send documents to MailRoom for further processing and transmission to other systems.

As MQSeries is operating on a buffer/structure concept of data and IMB is working with documents, it is necessary to represent the document in a large buffer with an agreed way to delimit each record in the document.

Using this method, single or multiple documents can be sent to the MailRoom.

Format

Before calling the MQPUT API, the business application must format the document into the MailRoom buffer format.

```
1 MQ_REC          UNALIGNED,
  3 HEADER,
    5 HDRTYPE      CHAR(01),
    5 DATATYPE     CHAR(01),
    5 CODEPAGE     CHAR(08),
    5 NBR_RECS     PIC'99999999',
    5 RESERVED     CHAR(82),
  3 DATA,
    5 .....     variable length depending on contents
```

Parameters

The fields in the structure are:

HDRTYPE
Format of the header part. Currently only one header is defined. Other headers might be added in the future to allow other representations of documents. Valid types are:

A Format as defined.

other Reserved for future use.

DATATYPE
Format of the detail data part. Valid types are:

L Each data-record is preceded by a two—byte binary length field. This representation is only usable if no ASCII to EBCDIC conversion is taking place (see also “Codepage translation services” on page 213).

```
<ll><data><ll><data> ... <ll><data>
               each <ll><data> to be interpreted as
                 5 LL      BIN FIXED(15),
                 5 DATA    CHAR(LL-2),
```

P Each data-record is preceded by a 4 byte character length field. This representation is good for transmissions between platforms with different codepages, where codepage conversion takes place.

```
<pppp><data><pppp><data> ... <pppp><data>
               each <pppp><data> to be interpreted as
                 5 PPPP    PIC'9999',
                 5 DATA    CHAR(PPPP-4),
```

other Reserved for future use.

CODEPAGE
Code page info for data (unimplemented at present)

ASIS	Data sent 'as is'. Base is EBCDIC.
blank	Data sent 'as is'. Base is EBCDIC.
other	Future use.

NBR_RECS
Number of (decoded) records. The data structure defined above is repeated NBR_RECS times.

RESERVED
Future use. Has to be initiated to blanks by calling application.

Format of input

- The data is passed to the MailRoom in the above buffer format.
- Notice that the length field preceding each data record is the length of the data record including the length of the length field depending on the value of DATATYPE.
- The input can contain multiple documents.
- An IMB MailRoom M-record, (either simple or extended) is used to indicate the beginning of each document. Other records must not have an "M" in column 1, as the record then would be misinterpreted as an M-record.
- The very first record must be an M-record.

Examples

Examples of how to format the buffer can be seen in Figure 14.

```
Input document :

MEXTREC00cccTO_TP                              FROM_TP
DKBAZBGM0 XXXXXXXXYYYYYYYYZZZZZZZZ
DKBAZNAD1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
DKBAZLIN7 BB0123

The first record must be a MailRoom M-record, the other records can
be any type/format not starting with 'M'.

buffer sent to MQ : <header><data1><data2><data3><data4>

Where    name    bytes    Contents
         <header> 100   = AP        00000004
         <data1>  304   = 0304MEXTREC00cccTO_TP  . . .
         <data2>   38   = 0038DKBAZBGM0 XXXXXXXXYYYYYYYYZZZZZZZZ
         <data3>   54   = 0054DKBAZNAD1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
         <data4>   20   = 0020DKBAZLIN7 BB0123
```

Figure 14. IMB MQSeries message format(s).

Using MQSeries MQGET to receive documents from MailRoom

Using a standard MQGET API call, an application on any supported MQSeries platform can receive documents from the MailRoom for further processing.

The structure of the received data is the same as described in “Using MQSeries MQPUT to send documents to MailRoom” on page 104, and it is the responsibility of the receiving application to split the MQSeries buffer into records depending on the specification in the header record.

Sending a Business Acknowledgment to MailRoom using MQSeries

Using a standard MQSeries PUT API call, an application on any supported MQSeries platform can send Business Acknowledgments to MailRoom to record the progress of a particular transaction through the business process.

The method used to send Business Acknowledgments to MailRoom is the same as sending a single or multiple documents to the MailRoom described in "Using MQSeries MQPUT to send documents to MailRoom" on page 104. As with documents a single or multiple Business Acknowledgment can be sent to the MailRoom.

Format

The format of Business Acknowledgment is described in "CICS MailRoom acknowledgment API—KBAXACP" on page 84 except that the KBAXACK record must be prefixed with a eight-byte field containing the value *BUSACK*.

Examples

Examples of the format of the buffer can be seen in Figure 15.

```
Input document :

BUSACK      400Q476100000000000001                  IKBBXX90001Process started
BUSACK      400Q476100000000000001                  FKBBXX90002Process finished

buffer sent to MQ : <header><data1><data2>

Where    name    bytes    Contents
         <header> 100   = AP        00000002
         <data1>  408   = 0408BUSACK      400Q476100000000000001           IKBBXX90001   Process started
         <data2>  408   = 0408BUSACK      400Q476100000000000001           FKBBXX90002   Process finished
```

Figure 15. IMB MQSeries message format of Business Acknowledgment.

Data conversion and MQSeries

When using MQSeries to transmit data between different platforms, there is a need for data conversion. IMB has been designed to use the standard function in MQSeries for data conversion. The use of MQSeries data conversion works differently in the MQ and SAP-MQ scenarios. The following describes the flow when sending and receiving data in the MailRoom.

SAP-MQ

Flow when using IMB to send data to SAP or receiving data from SAP. All data conversion takes place on the SAP side.

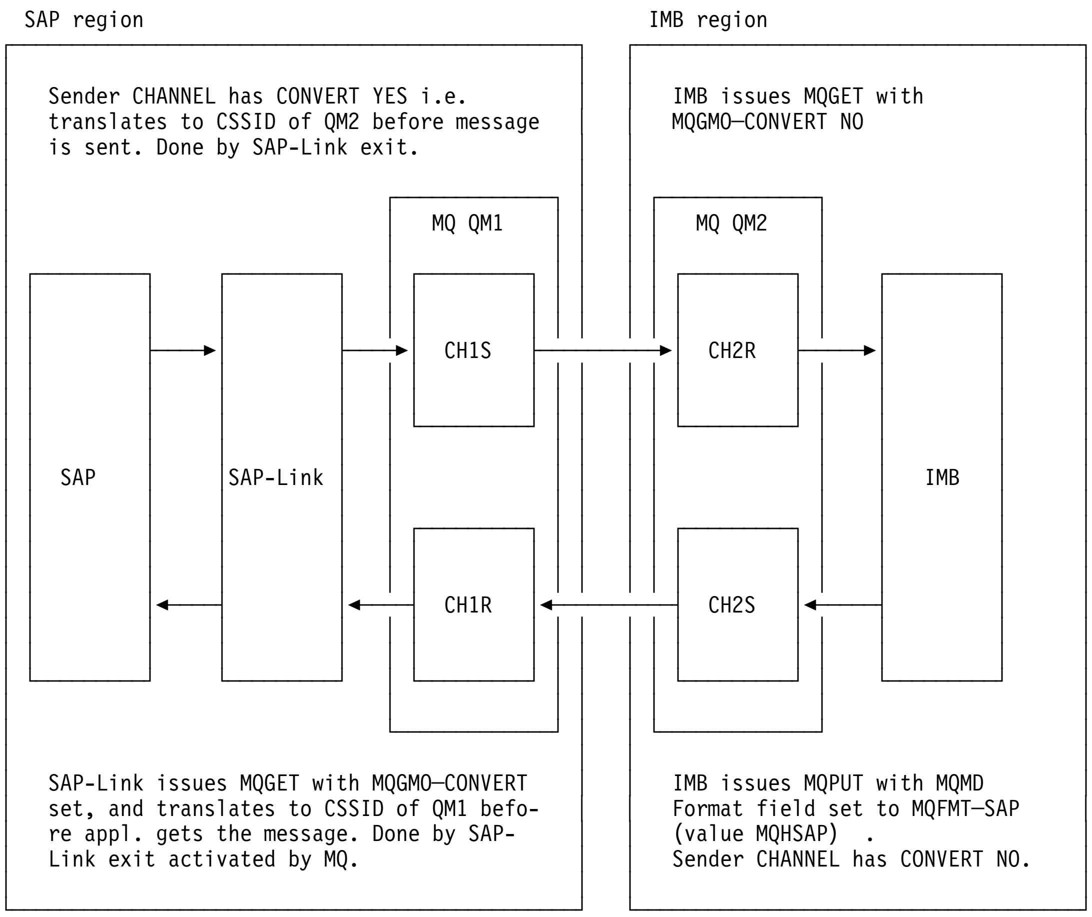
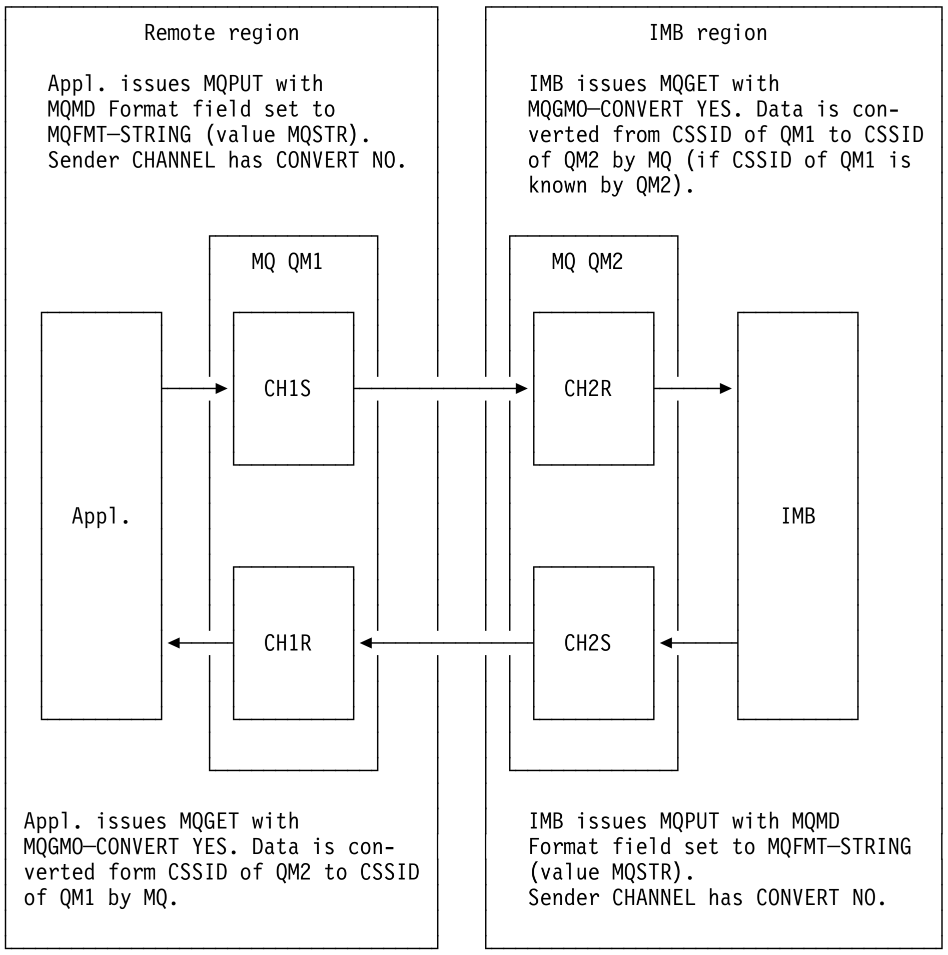

SAP region
IMB region
Sender CHANNEL has CONVERT YES i.e. translates to CSSID of QM2 before message is sent. Done by SAP-Link exit.
IMB issues MQGET with MQGMO—CONVERT NO
MQ QM1
MQ QM2
CH1S
CH2R
SAP
SAP-Link
IMB
CH1R
CH2S
SAP-Link issues MQGET with MQGMO—CONVERT set, and translates to CSSID of QM1 befo-re appl. gets the message. Done by SAP-Link exit activated by MQ.
IMB issues MQPUT with MQMD Format field set to MQFMT—SAP (value MQHSAP) .
Sender CHANNEL has CONVERT NO.

Figure 16. SAP-MQ

MQ

Flow when using IMB and MQSeries to transmit data to applications on another platform.


Remote region
Appl. issues MQPUT with MQMD Format field set to MQFMT—STRING (value MQSTR). Sender CHANNEL has CONVERT NO.
IMB region
IMB issues MQGET with MQGMO—CONVERT YES. Data is converted from CSSID of QM1 to CSSID of QM2 by MQ (if CSSID of QM1 is known by QM2).
MQ QM1
MQ QM2
CH1S
CH2R
Appl.
IMB
CH1R
CH2S
Appl. issues MQGET with MQGMO—CONVERT YES. Data is converted form CSSID of QM2 to CSSID of QM1 by MQ.
IMB issues MQPUT with MQMD Format field set to MQFMT—STRING (value MQSTR). Sender CHANNEL has CONVERT NO.

MQSeries definitions to access the MailRoom

To access the MailRoom using MQSeries some definitions are required to get data into the MailRoom.

The IMB MailRoom is running under control of CICS. To access the MailRoom using MQSeries the local MQSeries queue manager must be connected to the CICS system where the IMB MailRoom is installed.

MQSeries Queues

Passing data to MailRoom requires that one or more local queue(s) has been defined. There is no restriction to the name of the queue, but a few options have to be defined with special values to access the MailRoom. The definition of a local queue can be seen in the example below.

Notice the TRIGDATA option that must contain 'MQ' in position 1 to 6. If data is received from SAP-MQ Link the TRIGDATA option must contain *SAP-MQ* in position 1 to 6.

If the MailRoom has to handle data length fields in respect to DBCS shift-in and shift-out characters, the TRIGDATA option must contain *DBCS-CNV* in position 7 to 14.

In the example shown, an MQSeries queue “IMB.CICSID.MQ.INPUT” is defined:

```
DEFINE QLOCAL('IMB.CICSID.MQ.INPUT') +
         DESCR( 'Mailroom MQ source input Queue' ) +
         PUT( ENABLED ) +
         GET( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +
         TRIGGER +
         TRIGTYPE(FIRST) +
         TRIGDATA('MQ    ') +
         INITQ('IMB.CICSID.INIT') +
         PROCESS('IMB.PROCESS')
```

MQSeries Process

The process specified on the MQSeries Queue definition is used by MQSeries to decide which program that has to be started. The process has to point to transaction KBAG.

```
DEFINE PROCESS('IMB.PROCESS') +
         DESCR( 'Process for Triggers from MQ source Queue' ) +
         APPLTYPE (CICS) +
         APPLICID (KBAG)
```

MQSeries Initiation queue

MQSeries uses the initiation queue when the parameter TRIGGER is specified for the queue. The initiation queue must be activated by the CICS system using MQSeries CKTI (see the MQSeries for MVS/ESA System Management Guide for further information).

```
DEFINE QLOCAL('IMB.CICSID.INIT') +
         DESCR( 'Init Queue for IMB CICS' ) +
         PUT( ENABLED ) +
         GET( ENABLED ) +
         DEFPRTY( 5 ) +
         MSGDLVSQ(PRIORITY) +
         TRIGTYPE(FIRST) +
         NOTRIGGER +
         NOSHARE
```

MQSeries definitions to receive technical acknowledgement

To receive technical acknowledgement from MQSeries, a set of specifications is required to get it into the MailRoom.

These specifications are only needed for destination MQ, when ack.level 1 or 2 is specified, or destination SAP-MQ when ack.level 1 is specified. The definitions should only be used to receive technical acknowledgements. To receive Business Acknowledgements see “Sending a Business Acknowledgment to MailRoom using MQSeries” on page 107.

For further explanation of MQSeries specification and handling of reply queues see the *MQSeries Intercommunication Guide*.

MQSeries Reply Queues

Receiving technical acknowledgement from MQSeries requires that one or more local queue(s) has been defined. There is no restriction to the name of the queue, but a few options have to be defined with special values to access the MailRoom. The definition of a local queue can be seen in the example below.

In the sample below, an MQSeries queue *IMB.CICSID.MQ.ACK.REPLY* is defined:

```
DEFINE QLOCAL('IMB.CICSID.MQ.ACK.REPLY') +
         DESCR('Mailroom MQ reply Queue to receive technical ack') +
         PUT( ENABLED ) +
         GET( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +
         TRIGGER +
         TRIGTYPE(FIRST) +
         INITQ('IMB.CICSID.INIT') +
         PROCESS('IMB.ACK.REPLY.PROCESS')
```

MQSeries Reply Queue Alias

The queue specified on the receive subscription is defined as an alias to the local reply queue. The reason for this is the way that MQSeries replaces Queue Manager Name in its header information. The definition of a queue alias can be seen in the example below.

In the sample below, a MQSeries alias queue *IMB.CICSID.MQ.ACK.REPLY* is defined:

```
DEFINE QREMOTE('IMB.CICSID.MQ.ACK.REPLY.ALIAS') +
         DESCR('Mailroom MQ Reply Queue alias') +
         PUT( ENABLED ) +
         GET( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +
         RNAME('IMB.CICSID.MQ.ACK.REPLY') +
         RQMNAME('TRANSMITQ.MQ2.TO.MQIMB')
```

MQSeries Queue Manager Alias

The queue manager specified on the Reply Queue alias also has to be defined. The definition of a Queue Manager alias can be seen in the example below.

In the sample below, a MQSeries Queue Manager alias *TRANSMITQ.MQ2.TO.MQ1.IMB* is defined:

```
DEFINE QREMOTE('TRANSMITQ.MQ2.TO.MQIMB') +
         DESCR('Queue Manager alias for Mailroom MQ Reply Queue') +
         PUT( ENABLED ) +
         GET( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +
         RNAME(' ') +
         RQMNAME('MQIBM')
```

MQSeries Reply Process

The process specified on the MQSeries queue definition is used by MQSeries to decide which program that has to be started. The process has to point to transaction KBAB.

```
DEFINE PROCESS('IMB.ACK.REPLY.PROCESS') +
         DESCR('Process for Triggers from MQ Reply Queue') +
         APPLTYPE (CICS) +
         APPLICID (KBAB)
```

Chapter 6. MailRoom TIE/IMS support

The IMB MailRoom can be accessed from IMS platforms using TIE or TIE and MQSeries. This chapter describes the IMB MailRoom support for IMS platforms and the TIE definitions needed to access IMB MailRoom. It also provides guidance for selecting the appropriate method for the IMS installation (TIE alone or TIE with MQSeries).

Send documents to MailRoom
: How to use TIE to send documents to MailRoom.

Receive documents from MailRoom
: How to use TIE to receive documents from MailRoom.

Send Business Acknowledgment to MailRoom
: How to use TIE to send Business Acknowledgments to MailRoom.

TIE with MQSeries
: Overview of the standard TIE solution using ISClink.

TIE with MQSeries
: Overview of the TIE with MQSeries solution.

TIE or TIE-MQ?
: Considerations that apply when choosing between TIE and TIE with MQSeries, and how to convert your existing TIE transmissions to use MQSeries.

Handling error situations.
: What to do to retrieve data that has been read from TIE but could not be processed due to errors.

Using TIE-Write to send documents to MailRoom

Using a standard TIE-Write call (RWYWRIP) any business application can send documents to the MailRoom.

The structure of the data is agreed in each individual case. The business application should *wrap* the data in MailRoom format, adding a RIF with the M-Record format at the start of the data and a D prefix on all other records.

It is the responsibility of the business application to create the TIE Application and BTX, and define the RIFs which will be sent.

The owner of the business application must also ensure that mapping takes place to a TIE Recipient according to the TIE documentation, or that automatic re-route to another TIE is performed.

Once mapping is performed, TIE should re-route to IMS transaction KBAXTR0 for the TIE scenario or KBAXMR0 for the TIE-MQ scenario.

Using TIE-Read to receive documents from MailRoom

Using a standard TIE-Read call (RWYREAP) any business application can receive documents from the MailRoom for further processing.

The structure of the received data is agreed in each individual case. A MailRoom envelope consists of several records wrapped in MailRoom format (with an M-record as header and a D prefix on all other records). On transmission to TIE each record is assigned a RIF name based on the first 10 letters of the record after the MailRoom prefix D. The MailRoom TIE-Write program attempts to insert the RIFs it has built up to the TIE Application and BTX specified on the MailRoom service or subscription.

It is the responsibility of the business application to create the TIE Application and BTX, and define the RIFs expected. Any RIF created by the MailRoom TIE-Write program which is not defined in TIE will not be inserted.

Note: TIE has certain restrictions on the naming of RIFs. If the first 10 bytes of the record data (from which the RIF name is derived) contain characters that are not allowed in TIE RIFs, such as underscore or hex 00, the entire document will be rejected by TIE.

The owner of the business application must also ensure that a mapping takes place to a TIE Recipient according to the TIE documentation, or that automatic re-route to another TIE is performed.

Once mapping is performed, the business application can carry out the standard RWYREAP call to retrieve the data.

Sending a Business Acknowledgment to MailRoom using TIE

TIE can be set up to send Business Acknowledgments to MailRoom in the same way as original interfaces. The purpose is to record the progress of a particular transaction through the business process.

Format

A Business Acknowledgement consists of a single record containing the structure described in “CICS MailRoom acknowledgment API—KBAXACP” on page 84 and no M record is required. This record can be created in two ways:

- By mapping in TIE, for example to indicate that the document has reached TIE successfully and been passed on to the Business Application.

  In this case the values will either be taken from the KBAXDST RIF or supplied by constants.
- By the Business Application, either on receiving the document, on completing processing the document, or both.

For both of these methods the acknowledgment should be mapped to a recipient whose name begins with the string *KBAACK*. This step is essential as it identifies the transmission as an acknowledgment thus rendering the check for an M-record unnecessary.

The field SKBA_PROCESS_IND should be set to the relevant stage of processing depending on the required acknowledgment level specified in the ACKLVL field. The exception to this is where the Business Acknowledgment has been created by TIE mapping alone, in which case the field should be set to *Q*. When the TIE-Read API processes an acknowledgment with process indicator Q, it automatically converts this as follows:

- If the required acknowledgment level is 1, the document is considered finished upon receipt of this acknowledgment.
- If the required acknowledgment level is 2, MailRoom will wait for another Business Acknowledgment to be supplied by the Business Application.

  This second Acknowledgment may of course also be sent via TIE but the Business Application will fill the process indicator with the correct value and for this reason a second BTX will be required in TIE to provide this input.

Acknowledgment processing - TIE-IMS via ISCLink

On arrival in IMB, the Business Acknowledgment is processed by a different module from the one handling new documents. Although both types of transaction can use the same ISC link, it is necessary to set up a new entry in the WC table to specify the Business Acknowledgment program name.

Use the fastpath command **WCEDIT** to insert a *KBBEDAM* WC-table entry.

```
KBHWDMIM                Modify Working Criteria Entry

Country code . . . . . : 123

Key. . . . . . . . . . : KBBEDAM KBAAIRP
                         KBBEDAM xxxxxxxx

Value. . . . . . . . . . FKEIM04_________________________
                         11111111

Comment. . . . . . . . .
LTERM of ISC-Link (IMS-CICS)____________________________________________
```

Country code
The Trading Partner country

Key
The key the BEC module will use. KBBEDAM is the name of the BEC module reading this entry. KBAAIRP is the name of the MailRoom receiver module that BEC will invoke.

Value

Enter the LTERM name for the ISClink port between IMS and MailRoom region.

TIE-IMS Scenario using ISClink

This section describes the extra activities required when the link between TIE and the MailRoom is an ISClink.

IMB processing

On arrival in IMB, the document is processed by program KBASIRP.

Use the fastpath command **WCEDIT** to insert a *KBBEDAM* WC-table entry, to link this program to its ISClink.

```
KBHWDMIM                Modify Working Criteria Entry

Country code . . . . . : 123

Key. . . . . . . . . . : KBBEDAM KBASIRP
                         KBBEDAM xxxxxxxx

Value. . . . . . . . . . FKEIM04________________________
                         llllllll

Comment. . . . . . . . .
LTERM of ISC-Link (IMS-CICS)____________________________________________
```

Country code
The Trading Partner country

Key
The key the BEC module will use. KBBEDAM is the name of the BEC module reading this entry. KBASIRP is the name of the MailRoom receiver module that BEC will invoke.

Value

Enter the LTERM name for the ISClink port between IMS and MailRoom region.

Acknowledgments in the TIE-IMS scenario

Since IMB is responsible for the data until it has been written to TIE, an event will not be written until the TIE Write has been performed.

For acknowledgment level 1 and 2, please see “Sending a Business Acknowledgment to MailRoom using TIE” on page 118.

TIE-MQ Scenario

This section describes the extra activities required when using MQSeries with TIE to communicate with the MailRoom.

Sending documents to the MailRoom

There are two further requirements once the TIE and MailRoom definitions have been carried out.

- The DB2 table used to specify the MQSeries information
- The MQSeries definitions

The DB2 table KBDTMT must be created on the system where TIE is installed. For more details, refer to the *Installation Guide*.

At least one entry is required—the default entry with a key consisting of asterisks. The key is made up of the first three fields. It is also possible to use different queues according to which TIE application, recipient and country data arrives from. This gives the business application the possibility to send data to different IMB systems. It is recommended to use different queues for errors and data respectively in order to allow differentiation of the transactions in CICS.

This is an example of how the first five fields of the table might look (fields shortened for space reasons).

```
IOPUCTY ISYSIDY ITRNRCP    SKBA_MQQNAME         SKBA_MQQERR
------- ------- ---------- -------------------- --------------------
***     ****    ********** QUEUE.FOR.DATA       QUEUE.FOR.ERRORS
123     SYST    DATA       QUEUE.FOR.123.DATA   QUEUE.FOR.123.ERRORS
```

These table entries would result in the following:

- If an error is detected when reading data from the MailRoom, error messages will be sent to queue QUEUE.FOR.ERRORS.
- If data is sent to application SYST, BTX DATA in TIE, for country 123, it will be placed on queue QUEUE.FOR.123.DATA.
- If data is sent to application SYST, BTX DATA in TIE, for country 123, but errors are detected, the error message will be sent to queue QUEUE.FOR.123.ERRORS.
- Data sent to TIE for any other country, application or BTX will be sent to QUEUE.FOR.DATA, and any errors will be sent to QUEUE.FOR.ERRORS.

The MQSeries definitions are similar to the “MQSeries definitions to access the MailRoom” on page 110. However the TRIGDATA should be set to TIE-MQ.

If you are using a separate queue for errors then this queue should trigger CICS transaction KBAE.

If you are using TIE to transmit business acknowledgements please see “Sending a Business Acknowledgment to MailRoom using MQSeries” on page 107. In this case you should make a separate entry in the KBDTMT table pointing to this queue. The CICS transaction to trigger for business acknowledgments is KBAB.

Receiving documents from the MailRoom

The DB2 table described is used only for error messages. The only entry that will be used is the default entry with a key of asterisks. The Error queue must trigger transaction KBAE in MQSeries on the CICS side of the connection.

The MQSeries definitions on the IMS system where TIE is installed require that the queue uses IMS triggering. The IMS trigger function is a long-running BMP which checks the initiation queue regularly for input and inserts a transaction to IMS when data arrives. Set up the queue which is named on the MailRoom service or subscription and enable triggering using trigger type FIRST. The process specified on the MQSeries queue definition will be used to define the IMS transaction which is to be started. In the field APPLICID on the process you must therefore specify *KBAXMW0* which is the IMB TIE-Write for MQSeries module.

In the example, queue *IMB.CICSID.TIEMQ.OUTPUT* is defined with trigger type FIRST and the related process shows that IMS transaciton KBAXMW0 will be started.

```
DEFINE QLOCAL('IMB.CICSID.TIEMQ.OUTPUT') +
         DESCR( 'TIE-MQ Output from IMB' ) +
         PUT( ENABLED ) +
         DEFPRTY( 5 ) +
         DEFPSIST( YES ) +
         GET( ENABLED ) +
         TRIGGER +
         TRIGTYPE(FIRST) +
         TRIGDATA( 'MQ' ) +
         PROCESS( 'IMB.IMS.PROCESS') +
         INITQ( 'IMB.CICSID.IMS.INIT')
DEFINE PROCESS('IMB.IMS.PROCESS') +
         DESCR('Process for TIE-MQ from IMB' ) +
         APPLTYPE ('IMS') +
         APPLICID ('KBAXMW0') +
```

Acknowledgments in the TIE-MQ scenario

Since IMB is responsible for the data until it has been written to TIE, and there are two checkpoints underway, there are two acknowledgments returned to satisfy acknowledgment level 0.

Two events will be inserted for each document; one to say the data was written to MQ and the second to record the TIE Write.

For acknowledgment level 1 and 2, please see "Sending a Business Acknowledgment to MailRoom using TIE" on page 118.

TIE or TIE-MQ?

The TIE-MQ solution is the recommended way to transmit data from IMS platforms to MailRoom. The method has two advantages over the TIE-IMS method:

- It runs independently of the MailRoom.
- It is not required to use remote DB2 if the IMS and CICS installations are on different MVS systems.

Using TIE-MQ removes the need for ISC links and spares the IMS queue for undeliverable messages if the ISC link is unavailable.

If you have an existing TIE transmission to MailRoom and you wish to convert to MQSeries then you must perform the following actions:

- Change the TIE Recipient Operational Control to point to KBAXMR0 instead of KBAXTR0.
- Ensure the KBDTMT table has been created in the DB2 accessible to your IMS environment and set up an entry for the TIE Application, Recipient and country your data is sent from.
- Set up the MQSeries queues for data and error transmission. See "TIE-MQ Scenario" on page 121.
- Change the service or subscription from TIE-IMS to TIE-MQ source.

If you have an existing TIE transmission from MailRoom and you wish to convert to MQSeries then you must perform the following actions:

- Change the service or subscription from TIE-IMS to TIE-MQ destination; enter the MQSeries details.
- Set up the MQSeries queues for data and error transmission. See "TIE-MQ Scenario" on page 121.
- Set up an MQSeries report queue.
- Ensure the KBDTMT table has been created in the DB2 accessible to your IMS environment and set up a default entry in which the key fields are set to asterisks and the MQSeries queue for error messages is filled in.

Understanding error situations

This section describes how to retrieve input interfaces that could not be processed due to errors arising after the data had been read from TIE.

Likely error situations

The TIE source scenario differs from the other source scenarios in that a successful delivery from TIE to the IMB TIE-IMS or TIE-MQ transaction does not necessarily mean that data has been processed correctly. If, having read the TIE data in, the program is unable to process it, the data will not be saved automatically because the TIE-Read performs an IMS commit point. Typical situations preventing processing of the data could be DB2 errors or invalid MQSeries queue names.

In this situation, the TIE-Read and TIE-Read for MQSeries programs will perform a TIE-Write of the data to a special Error BTX in TIE.

Awareness of error situations

You might notice that source data is not present in IMB via a number of routes:

- The sender or recipient may notice data is not being processed.
- Alerts will be received due to errors in the ERRLOG.
- IMS Operational personnel may inform you that the IMB IMS transactions are abending repeatedly.

How errors are handled in the TIE scenarios.

If an error is encountered in the TIE-IMS scenario, and data has not yet been read from the IMS queue, the transaction abends and an error message is sent to transaction KBHELI0 for handling. If this transaction completes successfully the error message will be available in the ERRLOG transaction, otherwise it will be printed to SYSPRINT.

If an error is encountered in the TIE-MQ scenario, and data has not yet been read from the IMS queue, the transaction abends and an error message is sent to KBHELM0. This transaction loads the error message to the MQSeries queue named in the DB2 table and if this is successful it can be seen in ERRLOG. Otherwise the error message is written to SYSPRINT in the IMS region.

In both the above situations, which could be caused by a general DB2 error, the input data will remain in the IMS queue.

If data has already been read when the error is encountered, it is not possible to leave it on the IMS queue, so the data is written to the error BTX in TIE, and error reporting continues as above. If writing to the error BTX fails because the application or BTX have not been set up correctly, the data is lost.

Trouble-shooting

If you encounter a situation in which data is not reaching the MailRoom, and you are certain that it was processed correctly in TIE, you should perform the following steps:

1. Gather information.

- Look in the ERRLOG on IMB to see if any error messages have been logged regarding transmissions from TIE.
- If not, look in SYSPRINT for the IMS Region(s) in which the TIE-Read transactions run, to see if an error message has been issued. Check also whether the IMS Error handler transactions KBHELI0 and KBHELM0 are started. If not, start them and check the ERRLOG again.
- If no errors are found, display the IMS transaction that you expected to handle your data (specified in the BTX Operation Control in the TIE sending the data) and note whether it is stopped, and whether the queue count is greater than 0.
- If using standard TIE-Read, check whether the ISC Link between CICS and IMS is up.
- If using TIE-Read for MQSeries, check whether the MQSeries connection is operational, and whether the queues used for passing the data have a depth greater than 0.
- Look in the TIE Repository Monitor - Message Repository. If there is data to reprocess, messages will usually have been logged here.

2. Correct the error.
   - If a message was found in ERRLOG or SYSPRINT, take the appropriate action to correct the error, eg correct the plan, grant access, create the correct MQSeries queue or update the MQSeries details in the MQSeries information table. Continue at "Retrieve unprocessed data" below.
   - If the IMS transaction you were running is stopped, ensure any errors are corrected as above and restart the transaction. Continue at "Retrieve unprocessed data" below.
   - If using standard TIE-Read, and the ISC Link between CICS and IMS was down, restart the link. No further action is required.
   - If the message in TIE indicates that data was written to application STOP and that this BTX is stopped, continue at "Retrieve unprocessed data" below.
3. Retrieve unprocessed data.
   - If message KBAXTR005 or KBAXMR006 was found in the ERRLOG or SYSPRINT, the data could not be written back to TIE, and is lost. In this situation it is necessary to recover the data from the IMS log using the IMS message requeuer.
   - If message KBAXTR002 or KBAXMR005 was found in the ERRLOG or SYSPRINT, the data was written to TIE, and will be found in the Queued Input Repository (Table T016) in TIE. When you are sure that the error has been solved, list the BTX's belonging to TIE application STOP. Select the BTX which has the name BTXxxxxxxx, where xxxxxxx is the name of the transaction which failed. Display the BTX operational control for this BTX and locate the country number for which you have queued data. Change the S (stopped) to blank and press enter. Run the TIE Monitor Control Processor (this should be a BMP running constantly) to process the data. You may have to unstop other country entries for this and other BTX's if data from several sources has been written back to TIE. Using another session, ensure that further error messages are not being registered in ERRLOG or SYSPRINT. If this is the case you must change the STOP flag

back to S immediately and solve the problem before you can reprocess the data. **When all the data is processed, change the stop flag back to S. This last step is essential to prevent later loops occurring.**

- If message KBAXTR006 or KBAXMR002 was found in the ERRLOG or SYSPRINT, an error could have been detected before the data was read from TIE. In this situation, the IMS transaction will be stopped and the queue count greater than 0. It is normally sufficient to solve the error (eg DB2 connection failed, Plan access etc) and then restart the transaction. It is recommended that the Queued Input Repository (Table T016) in TIE is queried in this situation too, however, as one situation could mask the other in certain cases.

Chapter 7. MailRoom TCP/IP programs

The IMB MailRoom can be accessed from different platforms using TCP/IP. Two generic programs - T2F and F2T - are available on OS/2, AIX, Windows 95 and Windows NT. They can be used to send documents to, and receive documents from IMB MailRoom.

F2T - Send Program
Used to transmit documents to MailRoom.

T2F - Receive Program
Used to receive documents from MailRoom and put them into a file (and optionally invoke a program).

Sending Business Acknowledgments
Used to send one or more events to MailRoom about the progress of a received document.

AIX Scanner Program
Used to scan a directory for new files and then invoke a program (AIX only).

For the OS/2 platform two programs are available. They perform the same function as T2F and F2T, but they are more stand-alone with Presentation Manager windows and integrated directory scanning support.

OS/2 MailRoom Relay File2Tcp
Used to scan a directory for new files and transmit them to MailRoom

OS/2 MailRoom Relay Tcp2File
Used to receive documents from MailRoom and put then into a file.

TCP/IP MailRoom write/send program—F2T

The TCP/IP MailRoom Write/Send Program is a TCP/IP program to transmit documents to IMB.

Format

F2T is executable program available in versions for AIX, OS/2, Windows 95/98 and Windows NT.

The common call syntax is:

Call Syntax:

```
F2T, version: 1.00 TCP/IP connectivity with IMB, (C) Copyright IBM Corp. 2000

Usage: F2T FileName [ParameterFile]
  FileName:      Name of file to upload
  ParmeterFile:  Name of optional parameter file, default is F2T.dat

Format of parameter file:                                     Example values:
  HOSTNAME   = <HostName>   Name of host                      mvsx.yy.ibm.com
  PORTNAME   = <PortName>   Name of port on host              1812
  DELIMITOR  = <Delimiter>  Record delimiter when send        ##
  TRANSTAB   = <TransTab>   Codepage Translationtable         CP819500
  [CICSTRX]  = <CICSTrx>    Transaction in CICS               KBAJ or KBAL
  [ACTCODE]  = <ActCode>    Logon mode                        1, 2 or 3
  [ENCRYPT]  = <Encrypt>    Encryption of data                N or Y
  [COMPRESS] = <Compress>   Compression mode                  N
  [USERID]   = <Userid>     Userid, used for 2 and 3          XXUSER
  [PASSWORD] = <Password>   Password, used for 3              YYPASSWD
  [LOGFILE]  = <LogFile>    Name of optional log file         my.log
  [LOGLEVEL] = <LogLevel>   Level of logging                  INFO or ERROR
                            if not specified, ERROR is used.
```

Sample parameter file

The following is a sample parameter file F2T.DAT

```
F2T.DAT

HOSTNAME   = xxmvs.yy.ibm.com
PORTNAME   = 01812
DELIMITOR  = ¤¤
TRANSTAB   = CP819500
CICSTRX    = KBAJ
ACTCODE    = 1
LOGFILE    = upload.log
LOGLEVEL   = ERROR
```

More information can be found in the *READ ME* delivered with the code.

Usage of transaction codes in IMB:

- KBAJ - Used when sending business documents with M-record
- KBAL - Used when SAP R/3 IDOCs without M-record

TCP/IP MailRoom read/receive program—T2F

TCP/IP MailRoom Read/Receive Program is a TCP/IP daemon program listening to a port and waiting for incoming documents.

Format

T2F is an executable program available in versions for AIX, OS/2, Windows 95/98 and Windows NT.

The common call syntax is:

Call Syntax:

```
T2F, version: 1.00 TCP/IP connectivity with IMB, (C) Copyright IBM Corp. 2000

Usage: T2F [ParameterFile]
  ParmeterFile:  Name of optional parameter file, default is T2F.dat

Format of parameter file:                                      Example values:
  PORTNAME        = <PortName>   Name of port                  1812
  SAVEPATH        = <SavePath>   Where to store files          \IN
  [EXTENSION]     = <Extension>  File extension for files      DAT
  [FIRSTFILE]     = <FirstFile>  Filename to start with        1000
  [SAVEFILENAMES] = <YES/NO>     Save next free file name      NO or YES
  [PROCESSPGM]    = <ProcessPgm> Name of program to process a
                                 downloaded file.              myprog.exe
  [USERETCODE]    = <YES/NO>     Use return codes in program   NO or YES
  [LOGFILE]       = <LogFile>    Name of optional log file     my.log
  [LOGLEVEL]      = <LogLevel>   Level of logging              INFO or ERROR
                                 if not specified, ERROR is used.
```

Sample parameter file

The following is a sample parameter file T2F.DAT

```
T2F.DAT

PORTNAME      = 01812
SAVEPATH      = somewhere/download
EXTENSION     = idoc
SAVEFILENAMES = NO
PROCESSPGM    = somewhere/bin/my_prog
USERETCODE    = NO
LOGFILE       = download.log
LOGLEVEL      = ERROR
```

More information can be found in the *READ ME* delivered with the code.

Sending a Business Acknowledgment to MailRoom using TCP/IP

Using the MailRoom TCP/IP send programs, it is possible for a program to send Business Acknowledgments to MailRoom to record the progress of a particular transaction through the business process.

The method used to send Business Acknowledgment to MailRoom is the same as sending a single or multiple documents to the MailRoom described in "TCP/IP MailRoom write/send program—F2T" on page 128. As with documents a single or multiple Business Acknowledgment can be sent to the MailRoom.

Format

The format of Business Acknowledgment is described in "CICS MailRoom acknowledgment API—KBAXACP" on page 84 except that the KBAXACK record must be prefixed with a 8 bytes field containing the value *BUSACK*.

Examples

Examples of the format the buffer can be seen in Figure 17.

```
Input file:

BUSACK      400Q476100000000000001                  IKBBXX90001Process started
BUSACK      400Q476100000000000001                  FKBBXX90002Process finished
```

Figure 17. MailRoom TCP/IP document format of Business Acknowledgment.

AIX MailRoom scanner program—KBAUSCAN

The AIX Scanner Program looks for new files in a directory, and when anything is found invokes another program.

Format

KBAUSCAN is an AIX program, with the following syntax:

Call Syntax (AIX):

```
KBAUSCAN, version: V1R01M01

Usage: KBAUSCAN SleepTime ScanDir ReadyTime ProcessPgm [LogFile]

       SleepTime:      Sleep time in seconds between each scan

       ScanDir:        Directory that should be scanned
                       (Don't specify trailing slash)
       ReadyTime:      Ready time in seconds (the time a file has been
                       been ready for processing has not been touched)
       ProcessPgm:     Program that should be started with found file as argument

       [LogFile]:      Name of optional log file
```

OS/2 MailRoom write/send program—FILE2TCP

FILE2TCP is used to send files from the local TCP/IP host to a remote TCP/IP host (for example, IMB). When the program starts up, it will look in the directory <outbound_dir> immediately. If no file is found there, it waits for five seconds. Then it tries again. And again....

When a file fulfilling the search mask is found, it is read. All newline characters in the file are substituted by the <separator> string for example, ¢¢. Then the file is sent to the TCP/IP host stated in the parameter list (as a hostname). When the file has been sent successfully, FILE2TCP immediately checks if a new file is found in the <outbound_dir>. If this is so, this file is send also. If not, it waits for five seconds and tries again, and so on.

The program is closed by double clicking on the upper left corner (or other usual ways of closing Presentation Manager programs).

Format

FILE2TCP is an executable program only available for OS/2.

The call syntax is:

Call Syntax: (OS/2)

```
FILE2TCP <hostname> <portname> <outbound_dir> <finished_dir> <delimiter>
         <codepagetranslation> [<loglevel_information>] [<tracefilename>]
eg:

FILE2TCP MVSB 1719 D:\TCPRELAY\OUTBOUND\*.FOS D:\TCPRELAY\FINISHED\ ¢¢
CP850277 /LOGLEVEL ERROR TRACEFIL
```

More information can be found in the *READ ME* delivered with the code.

OS/2 MailRoom read/receive program—TCP2FILE

TCP2FILE is used to receive files from a remote TCP/IP host (for example, IMB). When the program starts up, it will immediately begin to listen on the TCP/IP port stated in the parameter list. When something is received, a check is performed that the format is reasonable. If so, all strings in the received text that are similar to the separator will be changed to new-line characters.

When the file transfer ends, the file is stored in the <inbound_dir>.

Format

TCP2FILE is an executable program only available for OS/2.

The call syntax is:

Call Syntax: (OS/2)

```
TCP2FILE <portname> <inbound_dir> [<file extension>]
        [<loglevel_information>]

eg:

TCP2FILE 1812 D:\TCPRELAY\INBOUND\ SAP
```

More information can be found in the *READ ME* delivered with the code.

Chapter 8. MailRoom APPC programs

The IMB MailRoom can be accessed from different platforms using APPC. Two generic programs, A2F and F2A, are available on OS/2, AIX, Windows 95 and Windows NT. They can be used to send documents to, and receive documents from IMB MailRoom.

F2A - Send Program
Used to transmit documents to MailRoom.

A2F - Receive Program
Used to receive documents from MailRoom and put them into a file (and optionally invoke a program).

Sending Business Acknowledgments
Used to send one or more events to MailRoom about the progress of a received document.

APPC MailRoom write/send program—F2A

The APPC MailRoom Write/Send Program is a APPC program to transmit documents to IMB.

Format

F2A is an executable program available in versions for AIX, OS/2, Windows 95/98 and Windows NT.

The common call syntax is:

Call Syntax:

```
F2A, version: 1.00 APPC connectivity with IMB, (C) Copyright IBM Corp. 2000

Usage: F2A FileName [ParameterFile]
  FileName:      Name of file to upload
  ParmeterFile:  Name of optional parameter file, default is F2A.dat

Format of parameter file:                                      Example values:
  SYMDESTNAME   = <SymDestName> Symbolic Destination Name       IMBCICS
  PARTNERTPNAME = <PartnerTP>   Name of the Partner TP          KBAC
  PARTNERLUNAME = <PartnerLU>   Name of the Partner LU          MPXXIMB
  MODENAME      = <ModeName>    Mode name for the session       CICSLU62
  DELIMITOR     = <Delimiter>   Record delimiter when send      ##
  TRANSTAB      = <TransTab>    Codepage Translationtable       CP819500
  [ACTCODE]     = <ActCode>     Logon mode                      1, 2 or 3
  [ENCRYPT]     = <Encrypt>     Encryption of data              N or Y
  [COMPRESS]    = <Compress>    Compression mode                N
  [USERID]      = <Userid>      Userid, used for 2 and 3        XXUSER
  [PASSWORD]    = <Password>    Password, used for 3            YYPASSWD
  [LOGFILE]     = <LogFile>     Name of optional log file       my.log
  [LOGLEVEL]    = <LogLevel>    Level of logging                INFO or ERROR
Note: If SYMDESTNAME is used, the TP, LU and Mode name parameters are ignored.
```

Sample parameter file

The following is a sample parameter file F2A.DAT

```
F2A.DAT

PARTNERTPNAME = KBAC
PARTNERLUNAME = MPXXIMB
MODENAME      = CICSLU62
DELIMITOR     = ¤¤
TRANSTAB      = CP819500
ACTCODE       = 3
USERID        = XXUSER
PASSWORD      = YYPASSWD
LOGFILE       = upload.log
LOGLEVEL      = ERROR
```

More information can be found in the *READ ME* delivered with the code.

APPC MailRoom read/receive program—A2F

The APPC MailRoom Read/Receive Program is a APPC Transaction Program activated by the communication software to receive incoming documents.

Format

A2F is an executable program available in versions for AIX, OS/2, Windows 95/98 and Windows NT.

The common call syntax is:

Call Syntax:

```
A2F, version: 1.00 APPC connectivity with IMB, (C) Copyright IBM Corp. 2000

Usage: A2F must be configured to be started by the Communication Systems
Attach Manager when inbound APPC requests arrives. It should be defined
as a Transaction Program with a parameter file:
  ParmeterFile:  Name of optional parameter file, default is A2F.dat

Format of parameter file:                                      Example values:
  SAVEPATH        = <SavePath>   Where to store files          \IN
  [EXTENSION]     = <Extension>  File extension for files      DAT
  [FIRSTFILE]     = <FirstFile>  Filename to start with        1000
  [SAVEFILENAMES] = <YES/NO>     Save next free file name      NO or YES
  [PROCESSPGM]    = <ProcessPgm> Name of program to process a
                                 downloaded file.              myprog.exe
  [USERETCODE]    = <YES/NO>     Use return codes in program   NO or YES
  [LOGFILE]       = <LogFile>    Name of optional log file     my.log
  [LOGLEVEL]      = <LogLevel>   Level of logging              INFO or ERROR
                                 if not specified, ERROR is used.
```

Sample parameter file

The following is a sample parameter file A2F.DAT

```
A2F.DAT

SAVEPATH      = somewhere/download
EXTENSION     = idoc
SAVEFILENAMES = NO
PROCESSPGM    = somewhere/bin/my_prog
USERETCODE    = NO
LOGFILE       = download.log
LOGLEVEL      = ERROR
```

More information can be found in the *READ ME* delivered with the code.

Sending a Business Acknowledgment to MailRoom using APPC

Using the MailRoom APPC send program, it is possible for a program to send Business Acknowledgments to MailRoom to record the progress of a particular transaction through the business process.

The method used to send Business Acknowledgment to MailRoom is the same as sending a single or multiple documents to the MailRoom described in "APPC MailRoom write/send program—F2A" on page 136. As with documents a single or multiple Business Acknowledgment can be sent to the MailRoom.

Format

The format of Business Acknowledgment is described in "CICS MailRoom acknowledgment API—KBAXACP" on page 84 except that the KBAXACK record must be prefixed with a 8 bytes field containing the value *BUSACK*.

Examples

Examples of the format the buffer can be seen in Figure 18.

```
Input file:

BUSACK      400Q476100000000000001                 IKBBXX90001Process started
BUSACK      400Q476100000000000001                 FKBBXX90002Process finished
```

Figure 18. MailRoom APPC document format of Business Acknowledgment.

Part 2. Gateway client/server support

IMB can act as a client/server gateway, where requests from clients are passed to the appropriate server after the necessary user authentication and authorization checks have been made.

The definition of such an external programming interface is called a Business Programming Interface (BPI). The required parameters of a BPI are:

- Name
- Type
- Program to run
- Program parameters
- Access type

Optional parameters are:

- ASCA logging to a file
- Limited opening time via a schedule

The access check can be one of these:

- Public, everyone with user ID on IMB can run the BPI
- Restricted, the access is controlled via IMB agreement sets

The BPI call is processed synchronously—the client will be in session with IMB while the request is security checked, passed to the appropriate local or remote server until the final reply is received.

Chapter 9. Client/server infrastructure

Overview of Gateway support

The infrastructure has been designed to be generic and extendable. Requests can be received in three ways:

- From a workstation or LAN server, using CIS-CSCS software
- From any platform using native LU6.2 (BEC protocol header)
- From CICS platforms using CICS Distributed Program Link (DPL)
- From platforms supported via IMB TCP/IP client API (this API is currently not generally available, but it is implemented in certain programs).

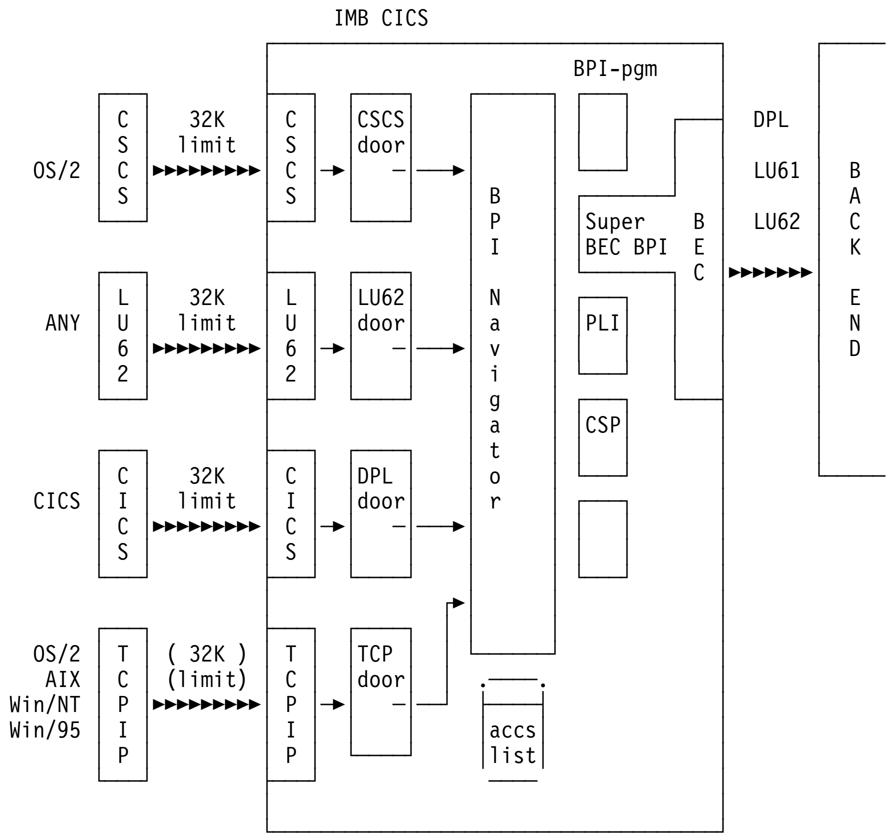


Figure 19. Client support

Requests received in three ways are mapped to the same internal structures and passed on to the BPI Navigator. Access authorizations are checked and the request is passed to the specified BPI module or to the generic BEC *Super BPI*.

Access to DB2

As the local BPI programs are executing under common infrastructure transaction codes, they also share the common DB2 Plan KBIA2AP. If the local BPI program needs to access DB2, the required DBRMs must be included in this plan.

The application DBRMs can either be included directly in the plan, or they can be bound into a DB2 Package, the latter method having the advantage that the common plan does not have to be rebound if the application package needs to be bound again.

The KBIA2AP plan includes only the DBRMs needed for use the IMB client/server infrastructure. If a locally developed BPI is using other parts of IMB, such as the MailRoom, the required DBRMs must be added.

Security

IMB uses the highest possible MVS and CICS security for the LU6.2 link between the client system and IMB, requiring a user to be signed on to the RACF system used by IMB for the link to be operational.

Additionally, the BPI Navigator has a facility to (optionally) log every message received and sent.

It is recommended to use this logging facility in case the application is sensitive (ASCA applicable).

Standard IMB Profiling query support

IMB has two public BPIs that can provide profile information about a user as well as passing the user's access list.

Both BPIs can be used to personalize the user interface for the currently logged-on user:

User Profile
: Information about the user-preferred language, Trading Partner number, country code (see “User profile BPI” on page 144).

Access List
: A list that can be used to disable or remove actions or options on menus, so that a user who is not authorized to use certain function will not be presented with those options when accessing the servers. See “Access list BPI” on page 145.

LAN Security administration philosophy
: This BPI can be used to implement a centralized administration philosophy.

  The resources defined in IMB do not necessarily have to define host based servers.

  By also defining LAN server resources using the IMB registration dialogues, a total server access control function can be implemented centrally for a project, and initialized for every user, individually during logon processing. This philosophy can drastically reduce application development and maintenance costs, as well as Service Delivery costs related to LAN server access administration.

Standard send structure

The standard send structure, as sent from the client, has this layout:

```
ISYSIDY    CHAR (04),       /* APPLICATION-SYSTEM ID      000 */
IBPIGRP    CHAR (08),       /* BPI GROUP NAME             004 */
IBPIFNC    CHAR (08),       /* BPI FUNCTION NAME          012 */
RESERV_16  CHAR (16),       /* RESERVED                   020 */
FUT_USE    CHAR (32),       /* RESERVED                   036 */
APPL_LNG   PIC'9999999',    /* LENGTH OF APPL_DATA        068 */
APPL_DATA  CHAR (var)       /* APPLICATION DATA           075 */
```

Send structure requirement

All strings must have the exact length specified and be padded with blanks at the end. You cannot use zero-terminated strings.

Standard receive structure

Data is returned to the client from IMB in this structure:

```
ISYSIDY    CHAR (04),       /* APPLICATION-SYSTEM ID      000 */
IBPIGRP    CHAR (08),       /* BPI GROUP NAME             004 */
IBPIFNC    CHAR (08),       /* BPI FUNCTION NAME          012 */
ENV_MSG    CHAR (10),       /* MESSAGE ID                 020 */
APPL_LNG   PIC'9999999',    /* LENGTH OF APPL_DATA        030 */
MSG_LNG    PIC'99999',      /* LENGTH OF MESSAGE TEXT     037 */
APPL_DATA  CHAR (var)       /* APPLICATION DATA           042 */
MSG_TXT    CHAR (var)       /* POTENTIAL MESSAGE DATA         */
```

BPI Navigator error messages

Apart from routing to the correct server, the BPI Navigator also has a security function. It checks whether the signed-on user ID is registered correctly in the IMB repository and is authorised to perform the requested function. If the request is rejected, an error message is returned by IMB to the calling client in its standard receive structure. Error messages from this module are prefixed with *KBI*.

There are three types of error messages:

- A technical message that gives information to developers (code AA)
- A Danish user version in Danish
- An English user version in code UK

The technical version (see Figure 20 on page 144). is not intended to be displayed to users. The version that the user receives depends upon the preferred language code for that user's IMB user ID profile.

An organization can support other languages by translating and inserting new rows in the error message (DB2) table.

```
KBIXXX -- - ----------------------------------------------------
KBIXXX AA I BPI infrastructure messages (AA - technical version)
KBIXXX -- - ----------------------------------------------------
KBI100 AA A User is not authorised to this BPI
KBI101 AA A Userid is not defined on IMB
KBI102 AA A The BPI is disabled
KBI105 AA A Unable to link to server or exit program.
KBI201 AA A Wrong rectype in call to BPI: KBH USER ACCESS
KBI202 AA A Wrong conversion mode passed to exit.
KBI203 AA A Wrong rectype in call to BPI: KBH USER PROFILE
KBI210 AA I Update/Refresh of KBDVBN was successful
KBI211 AA W No hits in KBIIBAP
KBI212 AA A Invalid function call to KBIIBAP
KBI996 AA A Internal program error. Error information = variable
KBI997 AA A Internal program error. Error information = variable
KBI998 AA A Not room for return message in receive area.
KBI999 AA A Wrong call-mode passed to query module.
```

Figure 20. Technical messages

If a message is returned, a message structure (KBIOMSG) is placed after any potential APPL_DATA. To extract such a structure: Start at the first byte of APPL_DATA, move to the right APPL_LNG bytes, and extract the next MSG_LNG bytes.

The MSG_TXT will contain a formatted message in the user' preferred language as defined in IMB. This is implemented using a *Multi language Message Server* in IMB see “Standard error message server” on page 154.

There can be an error message with or without application message data:

```
Example:
APPL_LNG = 327, MSG_LNG =   0  ====> no errors
APPL_LNG = 0,   MSG_LNG = 100  ====> no appl data, but one msg
APPL_LNG = 123, MSG_LNG = 100  ====> some appl data, and one msg
```

User profile BPI

IMB has implemented a standard, public server that can return the logged on user's profile, as registered in IMB.

The only *application data input* that the client must provide is a *response type indicator*, to tell the server in which layout the reply should be returned. The type has to be initialized to **01**.

(The standard header should specify the full, three level name of the BPI: KBH.USER.PROFILE)

The user ID need not be specified. It is the logged-on user ID, on behalf of which the client is executing.

Standard send structure data requirements for the Profile BPI

```
ISYSIDY    = 'KBH '
IBPIGRP    = 'USER    '
IBPIFNC    = 'PROFILE '
RESERV_16  = '                '
FUT_USE    = '                                    '
APPL_LNG   = '0000002'               (length of input APPL_DATA)
APPL_DATA:
  UP_REC_TYPE     CHAR(02);  = '01'  (requested profile type)
```

Response structure from the Profile BPI

The response returned to the client provides the data that is registered on the user's profile in IMB.

```
ISYSIDY    = 'KBH '
IBPIGRP    = 'USER    '
IBPIFNC    = 'PROFILE '
ENV_MSG    = '          '
APPL_LNG   = '0000084'
MSG_LNG    = '00000'
APPL_DATA:
  UP_REC_TYPE     CHAR(02)   = '01' (format of user profile reply)
  ZUSERID         CHAR(08)   = userid
  ZCUSRLAN        CHAR(02)   = preferred language code
  ZIOPUCTY        CHAR(03)   = user country code
  ZICUSPRM        CHAR(09)   = user's organization account number
  ZCUSTNAM        CHAR(60)   = name of user's organization
```

Access list BPI

IMB provides an application server that can return to the invoking client a list of the BPIs that the current user is authorized to use according to the IMB registrations. The name of the BPI server is KBH.USER.ACCESS

This server can be used to:

- Personalize the user's user interface in the client environment. For example, indicating in a pull-down menu an option that the user is not authorized to perform.
- Check whether a specific BPI (or group of BPIs) is available for processing or not (enabled or disabled), either during initialization of the user's environment, or immediately before sending a request to the server in question.

The length of the BPI input data is 22 characters: a BPI name and a *response type indicator*. The response can be provided in a short or long format, that is, with or without a 50 character description of the BPI.

Standard Send Structure data requirements for the Access list BPI

```
ISYSIDY    = 'KBH '
IBPIGRP    = 'USER    '
IBPIFNC    = 'ACCESS  '
RESERV_16  = '                '
FUT_USE    = '                                  '
APPL_LNG   = '0000022'               (length of input APPL_DATA)
APPL_DATA:
  ISYSIDY_SRCH    CHAR(04)   generic search on BPI system
  IBPIGRP_SRCH    CHAR(08)   generic search on BPI group
  IBPIFNC_SRCH    CHAR(08)   generic search on BPI function
  AL_REC_TYPE     CHAR(02)   '01'  (requested profile type, 01 or 02)
```

Generic search

is implemented using blanks. To get a list of all authorizations for this user for project *SWY*, the search criteria should consist of SWY followed by 17 blanks:

```
ISYSIDY_SRCH = 'SWY '
IBPIGRP_SRCH = '        '
IBPIFNC_SRCH = '        '
```

To get a list of ALL authorizations for this user, the search criteria should consist of 20 blanks:

```
ISYSIDY_SRCH = '    '
IBPIGRP_SRCH = '        '
IBPIFNC_SRCH = '        '
```

The requested response can be in a short (01) or a long (02) version as follows:

Response structure (short format) from the Profile BPI

```
ISYSIDY    = 'KBH '
IBPIGRP    = 'USER    '
IBPIFNC    = 'ACCESS  '
ENV_MSG    = ''
APPL_LNG   = '0000nnn'     nnn=5+AL_CNT*22
MSG_LNG    = '00000'
APPL_DATA:
2 AL_REC_TYPE     CHAR(02)   = '01' (short or long format)
2 AL_CNT          PIC'999'   = number of hits (max 100)
2 AL_ROW(100)                = repeated structure
  3 ISYSIDY       CHAR(04)   = BPI System
  3 IBPIGRP       CHAR(08)   = BPI Group
  3 IBPIFNC       CHAR(08)   = BPI Function
  3 CBPISCE       CHAR(01)   = BPI Security (1=Restricted,0=Public)
  3 CBPIENA       CHAR(01)   = BPI Enabled  (1=Yes,0=No)
```

Response structure (long format) from the Profile BPI

There is also a long version of the reply, where 50 char free text (description) and BPI-Type is also returned.

```
ISYSIDY    = 'KBH '
IBPIGRP    = 'USER    '
IBPIFNC    = 'ACCESS  '
ENV_MSG    = ''
APPL_LNG   = '0000nnn'     nnn=5+AL_CNT*76
MSG_LNG    = '00000'
APPL_DATA:
2 AL_REC_TYPE     CHAR(02)   = '02' (format of access list reply)
2 AL_CNT          PIC'999'   = number of hits (max 100)
2 AL_ROW(100)                = repeated structure
  3 ISYSIDY       CHAR(04)   = BPI System
  3 IBPIGRP       CHAR(08)   = BPI Group
  3 IBPIFNC       CHAR(08)   = BPI Function
  3 NBPITXT       CHAR(50)   = BPI Description
  3 CBPITYP       CHAR(04)   = BPI Type
  3 CBPISCE       CHAR(01)   = BPI Security (1=Restricted,0=Public)
  3 CBPIENA       CHAR(01)   = BPI Enabled  (1=Yes,0=No)
```

Optional BPI Navigator logging facility

The BPI Navigator has a facility to (optionally) log every message received and sent.

The log function stores the following information on a sequential file in the IMB start-up JCL.

BPI Navigator Log record

Field name	Type	Description
TIMESTAMP	Char(19)	CICS date and time
TASKNO	Char(8)	CICS tasknumber
TRANCODE	Char(4)	CICS transaction code
LUNAME	Char(17)	Fully qual.LUName (8+1+8)
USERID	Char(8)	RACF signed on userid
Ctrycode	Char(3)	Country code from userid profile
TPno	Char(8)	Trading Partner (Account) number from userid profile
reserved	Char(6)	Reserved for future use
Direction	Char(1)	*I* or *O* depending on direction
reserved	Char(5)	reserved for future use
Data	Char(var)	Application data. (complete standard send structure, including the BPI header)

Format of archive record optionally written by IMB.

When enabled, it logs certain information regarding every invocation of the BPI infrastructure, both on every incoming call as well as every outgoing reply.

Logging function transparently performed by the infrastructure, The logging (on or off) is decided during the IMB BPI registration dialogue.

Limit the availability of a BPI

A BPI is normally open all the time, unless it is disabled directly. If a BPI is accessing (or is running on) a back-end that has a more limited availability than IMB, then a schedule can be defined on the BPI. For information on how to define a schedule, refer to *System Administration Guide*.

A schedule can then be used on one or more BPIs, and the defined availability will then block the execution of the BPI program outside of the opening hours. The user will receive a generic message saying that the function is unavailable at present, or if the schedule has a specific user message defined, that one will be returned to the user.

Chapter 10. Client programming guidelines

Introduction

The IMB synchronous client/server Gateway can be accessed from three different types of clients:

- CICS systems (any CICS platform) using CICS DPL
- OS/2 systems using the CIS-CSCS software
- Any APPC-capable system using LU6.2

Access using CICS DPL

If the client environment is CICS, CICS DPL (Distributed Program Link) can be used. You can DPL to the IMB infrastructure (the *DPL door*) and IMB will then invoke the correct BPI, according to registrations.

To DPL to IMB from another CICS platform, this must occur:

- APPC connection to the IMB CICS
- Attach Security Verify (or Identify)

The normal IMB send and receive record interfaces are used.

The send structure is placed in 32K CICS communication area.

LINK to BPI interface on IMB:

```
EXEC CICS LINK
          PROGRAM('KBIDPLP')
          COMMAREA(COMMAREA)
          NOHANDLE;
```

The KBIDPLP program MUST be defined remotely in CEDA to point to the SYSID of the IMB CICS and use the remote transaction ID KBIM.

The standard receive structure is placed in same communication area by IMB, before returning control to caller.

Note: The DPL action from the Client environment is always to the IMB program KBIDPLP. The Business Application (BA) is the BPI that is registered as an IMB resource. The BPI is then invoked by the IMB infrastructure after authorization checking. The name of the BPI is passed from the client in the standard send structure to IMB. (See “Standard send structure” on page 143.)

Access using native LU6.2

The access to IMB BPIs from native LU6.2 is based on the native BEC protocol-modules and communication-modules.

The access to IMB from any platform using LU6.2 is based on the native BEC protocol and communication modules.

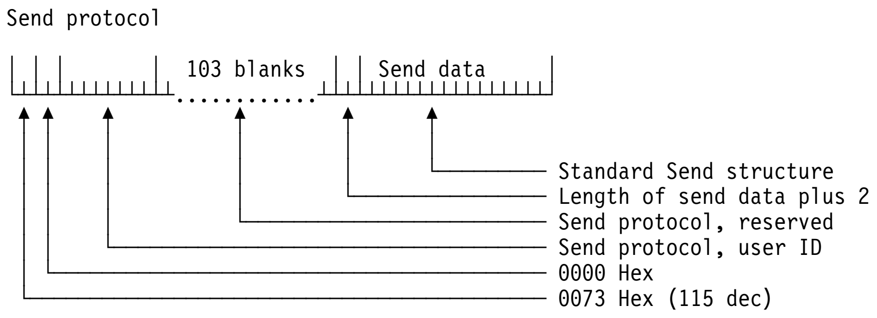
For the native LU6.2 option, the following should be in place:

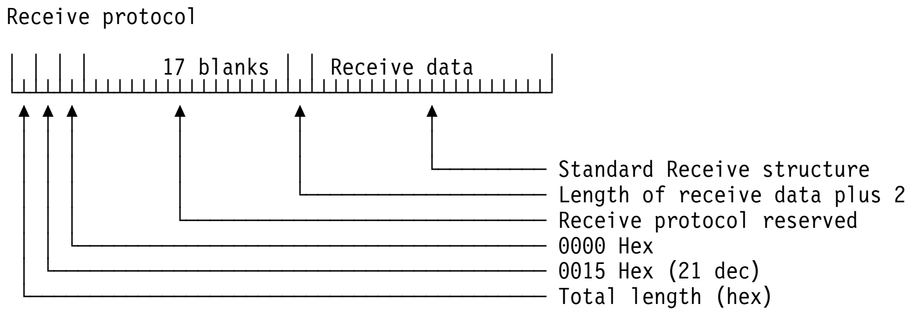
- APPC connection to the IMB CICS
- Attach Security Verify (or Identify)

The Standard send and receive structures are used.

However; the send structure should be prefixed by a two byte LL-field (HEX length of data including the two byte field itself)

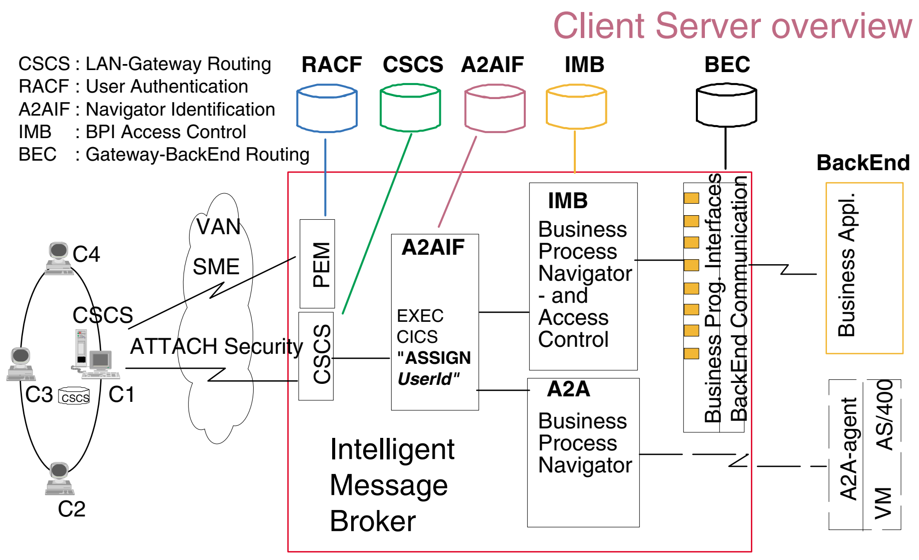
The receive structure is also prefixed by a corresponding LL-field.




Receive protocol
17 blanks
Receive data
Standard Receive structure
Length of receive data plus 2
Receive protocol reserved
0000 Hex
0015 Hex (21 dec)
Total length (hex)

Access using CIS-CSCS

Overview of connection



Client Server overview
CSCS : LAN-Gateway Routing
RACF : User Authentication
A2AIF : Navigator Identification
IMB : BPI Access Control
BEC : Gateway-BackEnd Routing
RACF
CSCS
A2AIF
IMB
BEC
BackEnd
C4
VAN
SME
CSCS
ATTACH Security
C3
CSCS
C1
C2
PEM
CSCS
A2AIF
EXEC CICS "ASSIGN UserId"
IMB
Business Process Navigator - and Access Control
A2A
Business Process Navigator
Business Prog. Interfaces
BackEnd Communication
Business Appl.
A2A-agent
AS/400
VM
Intelligent Message Broker

Figure 21. Access using CIS-CSCS

The connection between the LAN-CSCS and the Password Expiration maintenance (PEM) box at the host is to show that CSCS has implemented an OS/2 client application that is capable of communicating with the CICS/ESA PEM transaction. This is currently the only architected method of maintaining the password for a user, using the LU6.2 protocol.

Access from CIS-CSCS

When calling IMB BPIs using the CIS-CSCS protocol, the following should be in place:

- CIS-CSCS installed (locally or on LAN server)
- LU6.2 connection to the IMB CICS
- Attach Security Verify

The normal record interfaces are used.

The standard send structure must be prefixed with 8 character 'IBMMENU ' before calling CIS-CSCS.

Codepage translation is preformed on the workstation by the CIS-CSCS Server using an ASCII-EBCDIC translate table defined in CM/2. The translation is between the user's normal PC (ASCII) codepage and the user's normal host (EBCDIC) codepage. Data is then presented in the user's normal codepage (EBCDIC).

Access using TCP/IP

Using TCP/IP to connect to IMB BPIs is limited to programs written in C language in these environments:

- OS/2 (via a DLL)
- AIX (via object module)
- Windows NT and Windows 95 (via object module)

More information and sample programs can be found in *READ ME* files and in the header file *COMTCPIP.H* delivered with the code.

Chapter 11. Midlayer server programming guidelines

When the infrastructure receives a Client request, it first performs the authorization check to verify that the Client (user ID) is allowed to execute the named BPI. Assuming OK, it then invokes the BPI (EXEC CICS LINK), passing a commarea containing three pointers to the following data areas respectively:

```
DCL 1 COMM_AREA            BASED(COMM_AREA_PTR) UNALIGNED,
      2 ENVIRO_DATA_PTR     POINTER,
      2 CLIENT_DATA_PTR     POINTER,
      2 SERVER_DATA_PTR     POINTER;
```

The three pointers point to the following structures:

1. KBIENVR—*Environment data*
2. KBIIREC—*Input record*
3. KBIOREC—*Output record*

Standard BPI input and output structures

Irrespective of from where the Client request is originating, and which transmission protocol that has been used. The BPI navigator invokes the BPI program using the same three standard structures.

KBIENVR structure

The Environment record contains the static data that is registered in IMB for this BPI, as well as dynamic data about the current user related to this specific invocation of the BPI:

```
IUSRIDY    CHAR (08),       /* IMB user ID                 000 */
IOPUCTY    CHAR (03),       /* COUNTRY CODE               008 */
ICUSPRM    CHAR (09),       /* TRADING PARTNER NUMBER     011 */
CUSRLAN    CHAR (02),       /* LANGUAGE CODE              020 */
ISYSIDY    CHAR (04),       /* APPLICATION-SYSTEM ID      022 */
IBPIGRP    CHAR (08),       /* IMB BPI GROUP NAME         026 */
IBPIFNC    CHAR (08),       /* IMB BPI FUNCTION NAME 034      */
NBPITXT    CHAR (50),       /* IMB BPI DESCR. TEXT        042 */
CBPITYP    CHAR (04),       /* BEC OR OTHER TYPE          092 */
NBPIPGM    CHAR (08),       /* BPI PROGRAM                096 */
CBPISCE    CHAR (01),       /* BPI SECURITY               104 */
NBPIXIT    CHAR (08),       /* BPI DATA CONVERSION EXIT   105 */
CBPIENA    CHAR (01),       /* BPI ENABLE-DISABLE SWITCH  113 */
CBPIPM1    CHAR (20),       /* BPI PARAMETER NO. 1        114 */
CBPIPM2    CHAR (20),       /* BPI PARAMETER NO. 2        134 */
CBPIPM3    CHAR (20),       /* BPI PARAMETER NO. 3        154 */
CONV_MODE  CHAR (01),       /* TYPE OF CONVERSION (I/O)   174 */
BUILD_MSG  CHAR (01),       /* LET CALLER BUILD MESSAGE   175 */
MSG_ID     CHAR (10),       /* MESSAGE NUMBER             176 */
MSG_SUBST1 CHAR (25),       /* MESSAGE TOKEN 1            186 */
MSG_SUBST2 CHAR (25),       /* MESSAGE TOKEN 2            211 */
MSG_SUBST3 CHAR (25),       /* MESSAGE TOKEN 3            236 */
EXITPARM   CHAR (08)        /* PARM INFO TO EXIT          261 */
```

KBIIREC structure

The Client Input data is passed in the following structure:

```
ISYSIDY    CHAR (04),       /* APPLICATION-SYSTEM ID      000 */
IBPIGRP    CHAR (08),       /* IMB BPI GROUP NAME         004 */
IBPIFNC    CHAR (08),       /* IMB BPI FUNCTION NAME 012      */
RESERV_16  CHAR (16),       /* RESERVED                   020 */
FUT_USE    CHAR (32),       /* RESERVED                   036 */
APPL_LNG   PIC'9999999',    /* LENGTH OF APPL_DATA        068 */
APPL_DATA  CHAR (32692)     /* APPLICATION DATA           075 */
```

KBIOREC structure

The data that the BPI wants returned to the client should be passed to the Navigator in the following structure (pre-allocated by the infrastructure):

```
ISYSIDY    CHAR (04),       /* APPLICATION-SYSTEM ID      000 */
IBPIGRP    CHAR (08),       /* IMB BPI GROUP NAME         004 */
IBPIFNC    CHAR (08),       /* IMB BPI FUNCTION NAME 012      */
ENV_MSG    CHAR (10),       /* MESSAGE FROM IMB           020 */
APPL_LNG   PIC'9999999',    /* LENGTH OF APPL_DATA        030 */
MSG_LNG    PIC'99999',      /* LENGTH OF MESSAGE PART     037 */
APPL_DATA  CHAR (32725)     /* APPLICATION DATA           042 */
```

Standard error message server

This function is used *internally* by the IMB infrastructure. It is also available for application use—you can have IMB lookup messages for your own application if you add them to the IMB message table.

If the server upon returning control to IMB sets the BUILD_MSG field in the KBIENVR structure to *Y* and places a message ID in the MSG_ID field (in the same structure), then IMB will lookup the message in the table and return it to the client.

Three message variables &1, &2 and &3 in the text in the message table can be substituted at runtime with the contents of MSG_SUBST1, MSG_SUBST2 and MSG_SUBST3.

The MSG_TXT in the receive structure (see “Standard receive structure” on page 143) will then contain a formatted message in the user's preferred language as defined in IMB.

Alternatively you can decide to maintain all your application messages in the different languages on the LANs, in which case the server should handle the complete formatting of the KBIOREC itself.

The following is the layout of the 100 byte message area, as it is concatenated by IMB to the application data that is returned from the server. (See also “BPI Navigator error messages” on page 143.)

```
 MSG_TYPE      CHAR(02),      /* MESSAGE TYPE              000 */
 MSG_ID        CHAR(10),      /* MESSAGE ID                002 */
 MSG_TEXT      CHAR(88);      /* MESSAGE TEXT (INCL ID)    012 */
```

Super BPI exits

The BPI registration allows an exit to be defined for generic BPI programs (Super BPIs) for application specific purposes.

The exit program principle is currently utilized by IMB when the BPI-type is BEC (Super-BPI).

The exit is called both before and after calling BEC. Field CONV_MODE = I or O (Inbound/Outbound) tells the exit whether it can modify the Client data input or the Server data output.

The exit is linked to, with four pointers in the Commarea as follows:

```
DCL 1 COMM_AREA             BASED(COMM_AREA_PTR) UNALIGNED,
      2 ENVIRO_DATA_PTR     POINTER,
      2 CLIENT_DATA_PTR     POINTER,
      2 SERVER_DATA_PTR     POINTER,
      2 EXIT_PARMS_PTR      POINTER;
```

The EXIT_PARMS_PTR points into an additional record structure, which can be read or modified by the exit. The BEC Super BPI will pass a pointer to the BEC profile structure (record KBBPRINF).

This gives the exit access to a structure containing extra information from the Super BPI. The exit can then extract information and return it to the client.

The environment data field EXITPARM will contain the value *BEC PROF* in this case. The exit should only interpret the fourth structure if it can recognize the EXITPARM. This makes it possible to pass other structures in the future.

The exit can (on inbound) prohibit the real processing (here the call to BEC) by issuing an error message using the method described in “Standard error message server” on page 154.

BPI CSP modules

A BPI program can also be written in CSP. The application must be of the type *Called Batch*.

Under option **1 Application Specifications** in CSP the application must be type 4

```
Type of application => 4

  1  Main Transaction
  2  Main Batch
  3  Called Transaction
  4  Called Batch
```

The application will receive 3 working storage records, which are defined under option **4 Called Parameter List** in CSP. The application must have the following called parameters:

```
KBI__W_ENVR
KBI__W_IREC
KBI__W_OREC
```

The application can then read the input data from KBI__W_IREC record, read the PWS Environment data from KBI__W_ENVR record and finally place the result in KBI__W_OREC record.

The three records have the same layout as records for a normal BPI.

Chapter 12. Remote server programming guidelines

Historically, the BPI (the Server the Client is communicating with) is usually been implemented locally, on the IMB platform.

This extra piece of Business Application code was always needed on the mid-layer to perform data structure reformatting, to align to existing back-end applications.

As new projects emerge, with new Clients and Servers being coded, and therefore being aware of each other's data requirements, an option is needed to be able to invoke a (remote) server without any application code executing on the IMB platform.

For this purpose the BEC-BPI option is available. If the BPI is registered as type=BEC, the infrastructure can trigger a remote transaction, using the BEC protocol.

CICS DPL

It is not yet possible to use DPL to get to the application server/BPI—the data is passed using Pointers. A BPI type=DPL is possibly a candidate for a future IR.

Currently you must define a BPI type=BEC and have a set of BEC definitions for DPL, or you must develop a *generic* BPI type=CICS, that receives the application data pointed to by a Pointer, takes the data and moves it to its commarea and then DPLinks to the *real* server.

The effect of designing a solution this way is to trade some BEC registrations for some other CICS registrations.

Link to application server via the generic BEC-BPI

Using this *Super BPI* option, a Client request can be passed by IMB to the proper Server without any specific mid-layer programming. The following IMB registrations and restrictions apply:

BEC-BPI registrations

- The BPI should be of type BEC
- A BEC Appl Id is defined on BPI
- The country code of running user ID will be used as BEC Location
- BEC will perform the routing to BEC.
- Only APPL_DATA will be sent and received
- BEC length fields added

BEC-BPI data structures - example

If you need to send a structure containing the string **Hello World** from a PS/2 to an IMS program, using the CIS-CSCS protocol you could use the method described here.

On the PS/2 the string must be wrapped with an IMB Header structure. The header gives the logical name of the function to perform (the BPI Business Programming Interface) and a length field.

If the BPI is named XXX IMSACC HELLO and registered in IMB with a link to an IMS transaction (thru IMB back-end communication BEC) the following should be placed in the structure:

```
A2A_ROUTE   = 'IBMMENU '              /* A2A Routing info          */
ISYSIDY     = 'XXX '                  /* BPI System                */
IBPIGRP     = 'IMSACC  '              /* BPI Group                 */
IBPIFNC     = 'HELLO   '              /* BPI Function              */
RESERV_16   = ''                      /* Future use                */
FUT_USE     = ''                      /* Future use                */
APPL_LNG    = '0000011'               /* Length of APPL_DATA       */
APPL_DATA   = 'Hello World'           /* Actual data to send       */
```

The IMS program will receive the following structure:

```
header....
LL          = 13 (binary)             /* Length fld (incl itself)   */
MY_INP_DATA = 'Hello World'           /* Actual data                */
```

The reply from IMS program could be:

```
   ...protocol header....
LL          = 24 (binary)             /* Length fld (incl itself)   */
MY_OUT_DATA = 'Hello from IMS BackEnd' /* Actual data               */
```

The Standard Receive structure returned to the PS/2 program would look like this:

```
ISYSIDY     = 'XXX '                  /* BPI System                */
IBPIGRP     = 'IMSACC  '              /* BPI Group                 */
IBPIFNC     = 'HELLO   '              /* BPI Function              */
ENV_MSG     = ''                      /* No errors                 */
APPL_LNG    = '0000022'               /* Length of APPL_DATA       */
MSG_LNG     = '00000'                 /* No error struct after data */
APPL_DATA   = 'Hello from IMS BackEnd' /* Actual data received     */
```

If a message is returned (for example, if the user is unknown or has no access) a message structure (KBIOMSG) is placed after any APPL_DATA.

Part 3. NPT application design and development

This section describes NPT application design and development under IMB, and covers:

- Chapter 13, “NPT/3270 applications under IMB” on page 161 describes non-programmable terminal (NPT) (or 3270) applications under IMB.
- Chapter 14, “Intelligent Message Broker CSP application modules” on page 167 describes the IMB CSP application modules.
- Chapter 15, “Back-end programming” on page 183 describes back-end programming.

Chapter 13. NPT/3270 applications under IMB

While much of the information in this chapter concerns CSP application development, almost any CICS program can be started from IMB, requiring some programming effort to return the user to IMB after executing a foreign application.

CICS and CSP concepts

IMB is executing under control of CICS, and must therefore adhere to the principles of CICS programs and transactions. In CICS, a program can run in either *conversational mode* or *pseudo-conversational mode*.

In CSP the same concepts are called *nonsegmented mode* and *segmented mode*.

Table 12. CICS and CSP execution mode terminology

CICS	CSP	Description
Conversational	Nonsegmented	The resources are held during screen conversation.
Pseudo-conversational	Segmented	The resources are freed at screen conversation.

Conversational versus pseudo-conversational programs

Historically, storage was a limited resource and efficient program design was to use pseudo-conversational mode, because it uses less storage during execution in online environments. The storage is released during screen conversation while users are entering data on the terminal. Using conversational mode could fill storage with working storage from users who had left screens unattended, so storage allocation would be expended.

With today's CICS releases this is no longer a concern. The programmer can take advantage of the more structured and modular application design that conversational mode offers.

A pseudo-conversational program is a sequence of non-conversational programs ending by displaying a panel. Each program has to know the navigation path in order to guide the user back. Therefore it is no trivial task to reuse a program in another context.

A conversational program can call a program, and at some time the control will get back to the statement following the call. The called program does not know anything about the calling application, all it has to do is to be called with a well defined record and return with another. This leads to a more modular application design, where real reuse is a possibility.

IMB and underlying applications

IMB does not force underlying applications to run in particular execution mode. Both conversational mode and pseudo-conversational mode can be used. IMB Flexible Menu is running pseudo-conversational, while the administration applications are currently running in conversational mode. The design principles in this chapter also concentrate on the conversational mode. Some of the infrastructure components described in this chapter can be used by both conversational applications and pseudo-conversational applications, and some can not. Especially the support for navigation inside applications is limited to applications following the call principle.

In IMB we use the design principle *call of applications*.

This design supports:

- That activation of function key F3 results in a XFER/DXFR to *flexible menu* (KBHMEAP)
- That activation of function key F12 will result in *Converse Previous Panel*
- That input in the command field is split in a command part and a data part
  - The command is treated as a *fastpath command* (jump to another application)
  - The data is stored in KAAWCOM, from where they are available for processing in the *DXFRed to* or CALLed application

Multi Language Support implementation

When a new user is defined, a default Language Code will be stored in the user profile. This language code is a part of the key in the infrastructure tables to secure that messages and menus can be displayed in the user's preferred language.

The language code can be modified by the local administrator.

It is up to the country to decide which languages should be supported locally. When the IMB package is delivered from Denmark the only languages in the infrastructure are Danish (language code *DA*) and English (language code *UK*).

Table driven MLS

In IMB Multi Language Support is implemented by use of a DB2 table with all messages and CSP tables with all panel texts.

The panel texts must therefore be written in the supported languages at generation time while change of messages can be done online.

Online help can easily be accomplished via IMB help APIs (see “IMB online help system” on page 251).

Transaction change

All main applications are started by CICS using a transaction call.

The name of the transaction is registered in the option table when the Installer registers the option in IMB.

Each transaction must be defined to CICS to inform which application to start, and in RACF all users must be given access to the transactions.

Several applications can run in the same transaction as the main application (Stub application) can call other applications.

Each transaction has its own DB2 Plan which must be rebuilt if any SQL code has been changed in any of the applications in the transaction. Another approach is to use DB2 packages, where only the package must be rebound after code changes.

When a Stub application returns it must perform process KAAPDXF in order to ensure proper navigation.

Internal navigation

In IMB we control the internal navigation by use of the design principle *calls of applications*.

In this principle the navigation is controlled by CALLs to and EZECLOSEs from applications.

No maps (except for test maps - if any) are conversed in the main application. A called application is used for each map to be conversed. This way a panel can be reused, by invoking it with different function codes from different areas. IMB is widely using this idea to use same list application for both maintenance purposes and for prompt purposes.

When DXFR-ing from *flexible menu* to the main application the value assigned to ZGOTO of KAAWCOM determines which application to call. The initial ZGOTO value is found in the navigation table. In this way it is possible to use the same main application to support several options from flexible menu.

CSP applications which are the objects of a CALL statement must be defined as called applications and have parameters compatible with the CALL list defined for them. When the called application terminates execution, the called application resumes at the statement following the call.

The main application

When the main application receives control from *flexible menu*, the general process KAAPINI is performed to check the environment and to initiate KAAWCOM.

Figure 22 shows an example of the structure list for a main application.

```
NAME                LVL   OPTION    DESCRIPTION

KBGXSP_MAIN         001   EXECUTE   Mail Process
 KBGXSP_INIT        002   EXECUTE   Init
  KAAPINI           003   EXECUTE   Pre proces i each appl.
 KAAPDXF            002   EXECUTE   Navigate to other appl.
 KBGXSP_TEST        002   EXECUTE
  KAAPREC           003   EXECUTE   Pre converse proc; called appl
  KBGXSP_CONV_TEST  003   CONVERSE  Converse KBGXSMA
  KAAPCHC           003   EXECUTE   Post converse proc;called appl
    KAAPCMC         004   EXECUTE   Interpret command line
```

Figure 22. Structure list for main application

Figure 23 illustrates the fundamental structure of the main process of a main application.

```
PERFORM KBGXSP_INIT;
;
WHILE 1 = 1;
  ;
  IF ZGOTO ¬= '0';
    AND ZGOTO ¬= '1';
    AND ZGOTO ¬= '2';
    ;
    MOVE 'KBH105' TO ZMSGNO;
    MOVE 'IBMMENU' TO ZNEWAPPL;  /* internal command
    PERFORM KAAPDXF;             /* DXFR or XFER
  END;
  ;
  IF ZGOTO = '0';
    PERFORM KBGXSP_TEST;         /* Test purpose only (conv dummy-map)
  END;
  ;
  IF ZGOTO = '1';
    SET KBGXXW EMPTY;
    SET KBGXLWA EMPTY;
    ;
    MOVE ZICUSIDY TO KBGXXW.FLDOTHR1;
    MOVE ZICUSPRM TO KBGXXW.FLDOTHR2;
    MOVE ZCUSTNAM TO KBGXXW.FLDOTHR3;
    ;
    MOVE 'LOC' TO KBGXLWA.WHO;   /* local user adm
    MOVE 'LIS' TO KBGXLWA.WHAT;
    ;
    IF ZGO = 'GO';               /* Any parameters passed from cmd line?
      MOVE ZFIELDS(1) TO KBGXLWA.KBGXLI_FLDKEY1;
      MOVE ZFIELDS(2) TO KBGXLWA.KBGXLI_FLDKEY2;
      ....
      MOVE ' ' TO ZGO;
    END;
    ;
    CALL KBGXLAP KBGXXW,KBGXLWA,KAAWCOM;
    ;
    PERFORM KAAPDXF;             /* DXFR/ leave KAAAA ?
  END;
  ;
  IF ZGOTO = '2';
  ;
   ........
  ;
  END;
  ;
END;                             /* while
 ;
```

Figure 23. Main process of main application (Stub application). KAAWCOM, the general communication working storage record, is used as parameter in the call statement. The function code ZGOTO of KAAWCOM has been looked up in the navigation table before DXFR-ing from flexible menu. The ZFIELDS might contain arguments passed from the command line.

A main application post call process, KAAPDXF, is performed immediately after the call statement in the main application.

This system process will perform the navigation to other applications / transactions or return to the menu.

Called applications

Only called applications conversing maps are treated in this section.

In the called application parameter list the received parameters, KAAWCOM and others (if any) have to be specified.
Figure 24 shows the structure of the main process.

```
PERFORM KBGXLP_INIT;             /* Init application
;
WHILE 1 = 1;
  ....
  ;
  PERFORM KBGXLP_BUILD_MAP;      /* Build map - Working storage
  ;
  PERFORM KAAPREC;               /* Pre converse process
  ;
  PERFORM KBGXLP_CONV_MAP;       /* Show map
  ;
  IF EZEAID IS ENTER;
    OR EZEAID IS PF3;
    OR EZEAID IS PF12;
    PERFORM KAAPCHC;             /* Post converse process
  END;
  ......
  ;
END;
```

Figure 24. Main process structure in called application

Message handling

Messages can be retrieved from the message table by the CSP application KAAAMSG. They are identified by a message number, which must be assigned to to field ZMSGNO. KAAAMSG should be called by a pre pre-converse process like KAAPREC.

In order to maintain a general structure KAAAMSG should always be used to retrieve messages.

All types of messages should be handled the same way and with the same steps:

1. Somewhere in a process a message number is assigned to ZMSGNO
2. When KAAPRE receives control the corresponding message is retrieved in the message table by KAAAMSG who places the message in ZMSG. This field should therefore be included as message line on all CSP maps.
3. When KAAPCHK receives control all fields involved in the message processing are cleared.

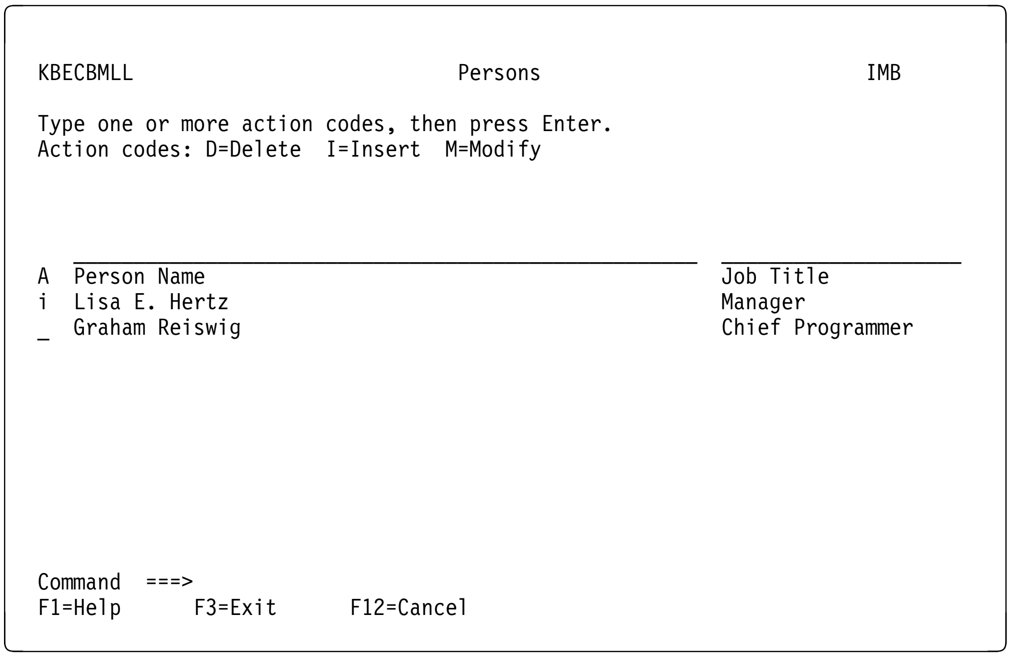
Chapter 14. Intelligent Message Broker CSP application modules

A typical NPT task in IMB is to show and update a list of some information taken from DB2 tables.

An example is the insertion of persons which— a Local Administrator task.

Task: Insert the person *Mr. Steven Levenson*, a clerk who has the job title *FOS Operator* and telephone extension number 6276.

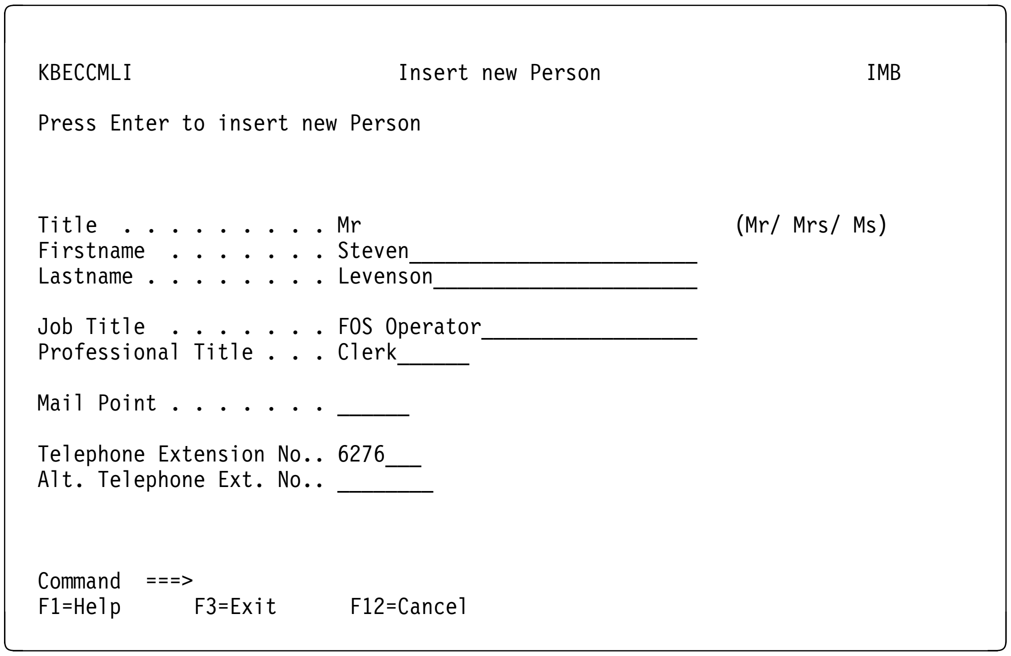
Navigate to the Persons menu using fastpath command **PERSON**.



```
KBECBMLL                           Persons                          IMB

Type one or more action codes, then press Enter.
Action codes: D=Delete  I=Insert  M=Modify



A  Person Name                                                  Job Title
i  Lisa E. Hertz                                                Manager
_  Graham Reiswig                                               Chief Programmer








Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 25. Persons panel

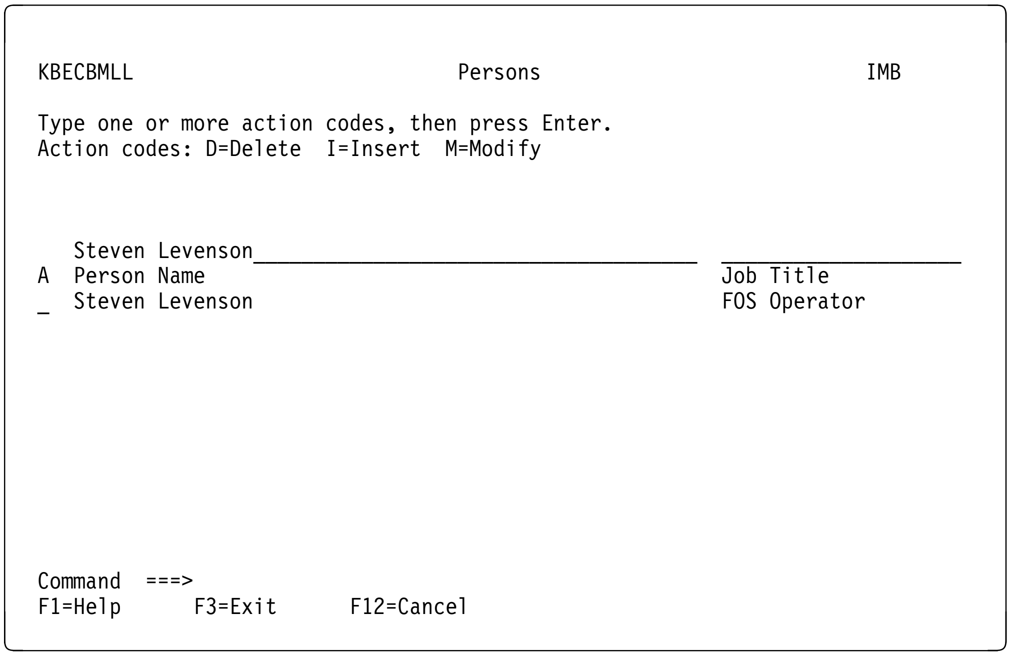
Type action code *i*, insert new person, in any of the action fields.



```
KBECCMLI                      Insert new Person                      IMB

Press Enter to insert new Person



Title  . . . . . . . . . Mr                             (Mr/ Mrs/ Ms)
Firstname  . . . . . . . Steven________________________
Lastname . . . . . . . . Levenson______________________

Job Title  . . . . . . . FOS Operator__________________
Professional Title . . . Clerk______

Mail Point . . . . . . . ______

Telephone Extension No.. 6276___
Alt. Telephone Ext. No.. ________



Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 26. Insert new Person panel

Complete the information about Mr. Steven Levenson and press enter.



```
KBECBMLL                           Persons                           IMB

Type one or more action codes, then press Enter.
Action codes: D=Delete  I=Insert  M=Modify




   Steven Levenson_____________________________________  ____________________
A  Person Name                                           Job Title
_  Steven Levenson                                       FOS Operator










Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 27. Adding a new Person

In the Persons menu the new person is displayed.

From a developer standpoint, this task can be broken down to these tasks:

1. Navigate to the application showing the list of persons.
2. Get the list of persons to show on the panel from DB2.

3. Show the list and navigate on action. Here action *I* requests navigation to an *insertion* application.)
4. Show the insertion panel, and validate the input.
5. Insert the Person in the DB2 table.
6. Re-display list with the new person.

Skeletons are provided for each of these functions.

Navigation (Stub applications)
The Installer defines which application to execute when the user chooses an option. IMB passes a function parameter to the application so it can select which List application to call depending on the function parameter. For more information refer to the *System Administration Guide*.

List applications
The main tasks for a List application is to show a list meeting the search criteria entered, to perform scroll logic and to navigate according to the user input. The user can use Function keys or enter action codes next to a line.

The data to be shown on the list is fetched by a call to a Browse application.

Browse applications
A Browse application is called by a List application and makes the necessary SQL call to DB2 tables to return a number of rows meeting the search criteria and the scroll keys given in the call.

Detail applications
A detail application is called from a List application to insert, modify, delete or show detail information. In the first three cases one or more DB2 tables are to be changed, after the input has been validated. The updating of DB2 tables is done by calls to updaters, one per DB2 table.

Update applications
Are called from Detail applications. An Updater creates the SQL necessary to insert, delete or modify a single row in one DB2 table.

In Figure 28 on page 170 the application flow is shown.

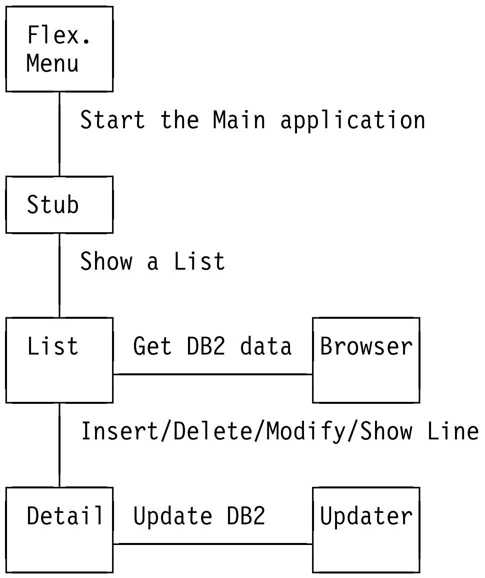


Figure 28. Application flow

The application types are described in this chapter.

Stub applications

An IMB Stub application is a CSP application started by IMB.

A Stub is defined with Type of application = Main Transaction and is used to call an underlying List application.

Before starting a Stub application IMB moves a function parameter to the field ZGOTO in KAAWCOM. This value is specified as the function code in the definition of the options and is used by the Stub to decide which List application to start.

Before the Stub call the List application, some parameters should be initiated as shown here:

```
MOVE 'IBM' TO KBEIBWA.WHO;   /* IMB installer
MOVE 'LIS' TO KBEIBWA.WHAT;  /* List panel mode
```

These parameters are used by the List application to decide which action codes are to be shown.

A Stub application can be generated automatically with the IMB Delevopment skeletons (available on request). The generated code will need minor modification before use.

List applications

An IMB List application is an application called by a Stub application, another List application or a Detail application.

The module is a called CSP application.

Call Syntax (CSP)

```
►►──CALL sssaaAP sssaaWA, KAAWCOM;──────────────────────────────►◄
```

Here the Working Storage record sssaaWA is used for specific information to the List application about function and search criteria. KAAWCOM is the common IMB work area.

First, the List application gets data to be shown on the list by a call to a browser developed for this purpose. The search parameters from the record sssaaWA are used as input to the browser together with information like number of lines on the panel list.

List applications presents the data on a map together with information about valid Action codes and F-Keys and makes the appropriate calls to Detail applications or others List applications.

As the List application uses the information from the Browser in the call to Detail applications more information than shown on the map is needed.

If the user wants to delete or modify the row timestamp has to be sent to the Update application and therefore first to the Detail application.

Often all information can not be written on one line, in this case the List application shows the action code *S* to let the user select the row for detailed information in a Detail panel.

All the information shown in the Detail panel must be sent by the List application in the call.

Multiple Function Support

Often the same list is to be shown in different situations demanding different actions on the list.

One example is the Administration part of IMB where both a Central Administrator and a Local Administrator can see the same list, but are allowed different actions on the items on the list.

Another example is the need to use the list to select an item to a field in a detail panel.

To ensure that the same List application can manage different functions we have introduced two control parameters in sssaaWA, WHO and WHAT.

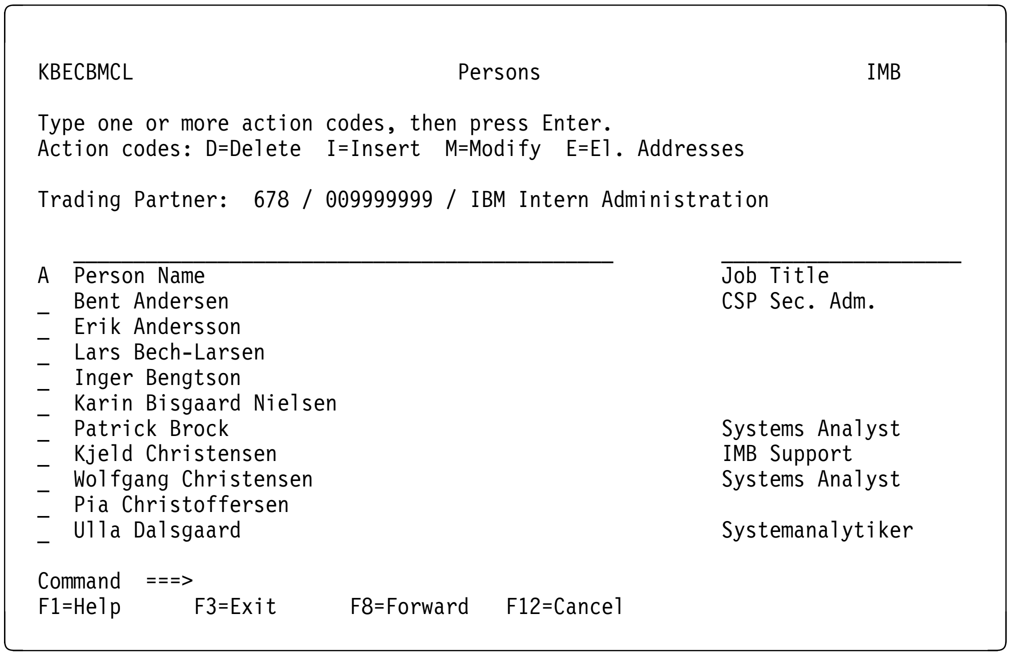
WHO tells who the user is (Central Administrator or Local Administrator) while WHAT tells what function the calling application wants.

The skeleton List application is built with two functions:

List mode.
A List application called from a Skel application or from another List application has the value *LIS* in sssaaWA.WHAT which shows the list panel in List mode.

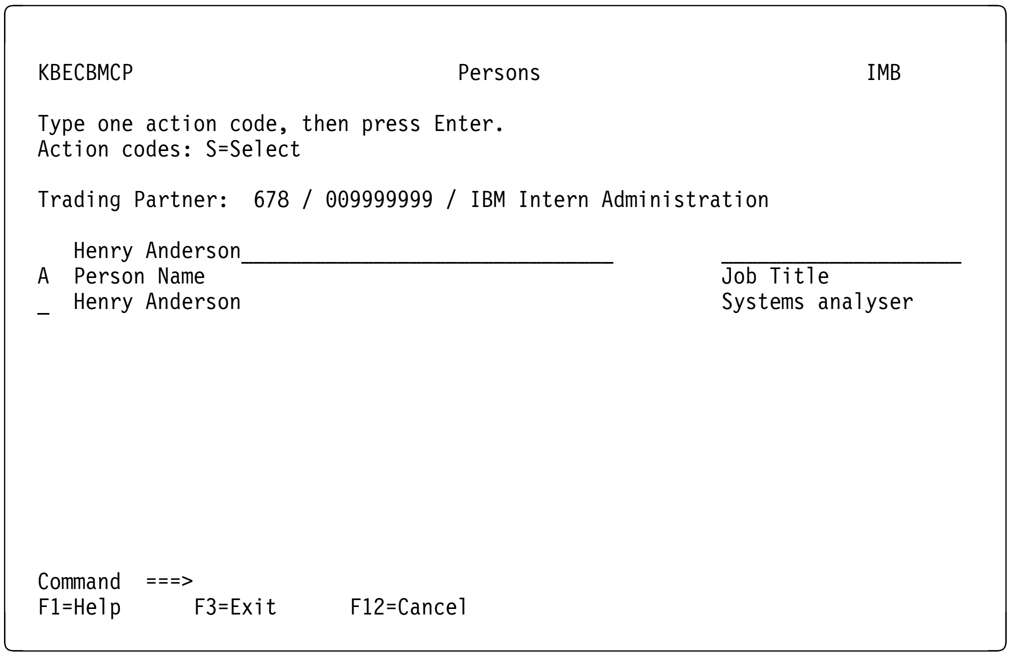
Prompt mode.
A List application called from a Detail application has the value 'PRO' in sssaaWA.WHAT. This means that the list is shown in prompt mode.

Figure 29 and Figure 30 show the same List application in the two different modes:



```
KBECBMCL                           Persons                          IMB

Type one or more action codes, then press Enter.
Action codes: D=Delete  I=Insert  M=Modify  E=El. Addresses

Trading Partner:  678 / 009999999 / IBM Intern Administration

   ______________________________________________         ____________________
A  Person Name                                            Job Title
_  Bent Andersen                                          CSP Sec. Adm.
_  Erik Andersson
_  Lars Bech-Larsen
_  Inger Bengtson
_  Karin Bisgaard Nielsen
_  Patrick Brock                                          Systems Analyst
_  Kjeld Christensen                                      IMB Support
_  Wolfgang Christensen                                   Systems Analyst
_  Pia Christoffersen
_  Ulla Dalsgaard                                         Systemanalytiker

Command  ===>
F1=Help      F3=Exit      F8=Forward   F12=Cancel
```

Figure 29. List panel (List)



```
KBECBMCP                           Persons                          IMB

Type one action code, then press Enter.
Action codes: S=Select

Trading Partner:  678 / 009999999 / IBM Intern Administration

   Henry Anderson________________________________         ____________________
A  Person Name                                            Job Title
_  Henry Anderson                                         Systems analyser









Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 30. List panel (Prompt)

Multi Language Support (MLS)

IMB has Multi Language Support.

A List application tries to find the texts for the panel in the language preferred by the user (ZCUSRLAN in KAAWCOM). If this language is not supported, English is used instead.

CSP Tables

A List application uses three CSP tables that must be changed by the programmer:

sssaaTP - Function key table
This table holds information of the F-key text in the supported languages. The text is found by the List application using the language code as key.

sssaaTA - Action code table
This table holds information of the legal action codes and their leading texts in the supported languages. The key is built like LLHHHWWWI where

- LL is the language code
- HHH is who (for example, CEN for Central Administrator)
- WWW is what (for examplem, LIS for List)
- I is the action code

sssaaTB - Panel text table
This table holds information of the text for the items on the map. The key is built in the for LLHHHWWWIIIIII where:

- LL is the language code (for example UK for English)
- HHH is WHO (for example CEN for Central Administrator)
- WWW is WHAT (for example LIS for List or PRO for Prompt)
- IIIIII is the item identifier (for example ACTION for the action codes)

A List application can be generated automatically with the IMB Delevopment skeletons (available on request). The generated code will be almost ready to use, but will need some modification, especially to the screen layout.

Browse applications

An IMB Browse application is an application called by a List application to get data from DB2 tables to be shown on a list panel.

The module is a CSP application with the following CALL syntax:

Call Syntax (CSP)

```
►►──CALL sssnnAP sssnnWP5,sssnnWP6 (NOMAPS;──────────────────►◄
```

The Working Storage record sssnnWP5 is the input record and sssnnWP6 is the output record.

The input record contains these items:

QROWCNT - Row count
: Tells how many rows the List application wants returned. This is the number of rows that can be shown on the list panel.

SCRLLDIR - Scroll direction
: If SCRLLDIR = 'F' then the browser finds the rows after the last SCRLLKEY, else SCRLLDIR = 'B' and the browser returns the rows before the first SCROLLKEY.

APPLNAM - Application Name
: Name of the calling application.

USERID
: Name of userid logged on.

sssnnI_DB2ROW
: A DB2 row used for search parameters.

SCRLLAREA
: Contains the top and bottom row last shown on the list panel. Used for scrolling purposes.

The Browser makes the SQL queries against the necessary DB2 tables to return the number of rows asked for. One extra SQL query is then made to return if more row(s) are found. If the scroll direction is 'B' and not enough rows can be found then the scroll direction is changed and rows are taken from the top.

Any serious SQL errors are logged to the System Errorlog.

The Browser returns the following in the output record sssnnWP6:

MSGNO - Message Number
: A code to an IMB message if something went wrong.

RETCODE - Return code
: RETCODE = 3 if scroll direction has been changed.

SQLCODE - SQL Error Code.
: SQL Error Code.

MORE
: MORE = 'Y' if more rows can be found in the scroll direction.

TABLEID - Table ID
: ID of the DB2 table used. If more than one table is used TABLEID contains ID of one of the tables.

QROWCNT - Row count
: The number of rows returned. Less or equal with QROWCNT in the input record.

SCRLLDIR- Scroll Direction
: Scroll Direction used (can be different than SCRLLDIR in the input record).

sssnnI_DB2ROW
: The returned rows. Max 20 rows.

SCRLLAREA
: The upper and lower returned row. (This must be moved to the input record at next call.)

A Browse application can be generated automatically with the IMB Delevopment skeletons (available on request). The generated code will only need a little touch in the SQL before it is ready to use.

Detail applications

An IMB Detail application is an application called by a List application to perform one of these actions:

- Show more information about a row from the list than can be seen on one line.
- Insert a new item on the list.
- Modify an item on the list.
- Delete an item from the list.

Call Syntax (CSP)

```
►►──CALL sssbbAP sssbbWA, KAAWCOM;─────────────────────────────────►◄
```

The Working Storage record sssbbWA is used for specific information to the Detail application about function and data from the selected row and a return code from the Detail application. KAAWCOM is the common IMB work area.

The record sssbbWA contains the following items:

WHO Input parameter WHO tells who the user is (Central Administrator or Local Administrator).

WHAT
Input parameter. The function to be performed by the Detail application. (Select / Insert/ Modify / Delete / Copy)

CANCEL
Return parameter. Is 'YES' if the user has left the Detail application by pressing F12.

CALL
No longer used.

PFVALID
No longer used.

GO No longer used.

sssbbI_DB2ROW
This is the data from the row on the List application not only the data seen on the list panel but all the information returned from the DB2 tables inclusive a timestamp. (If the action is Insert sssbbI_DB2ROW is empty)

The Detail application presents the data on a map together with information about valid Action codes and F-Keys.

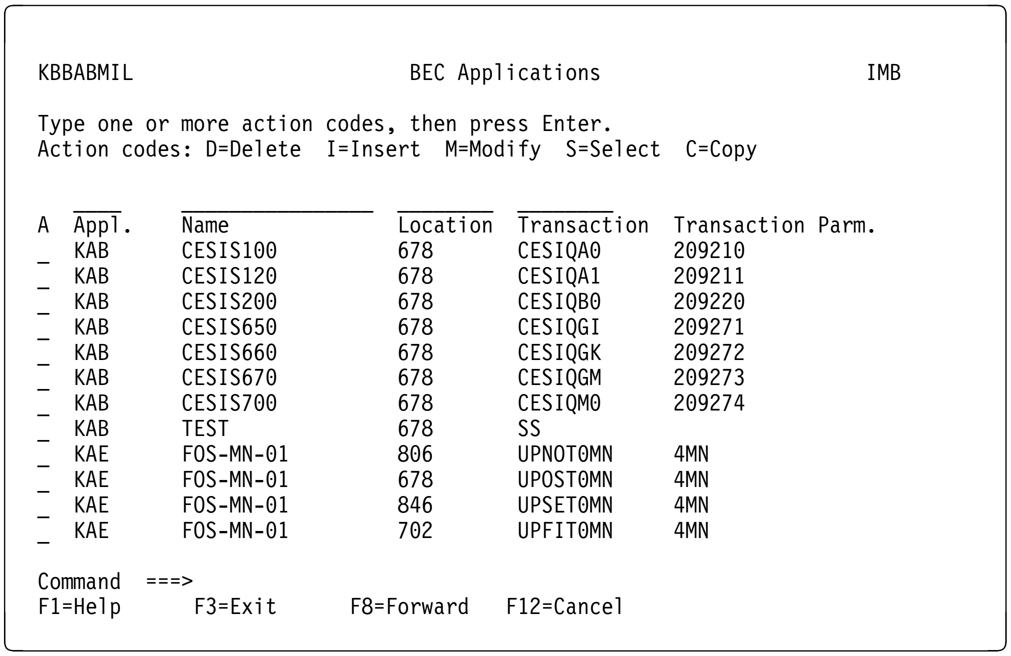
Multiple Function Support

We have built the Detail application to support all the actions on a list panel line not resulting in another list panel.

The functions are:

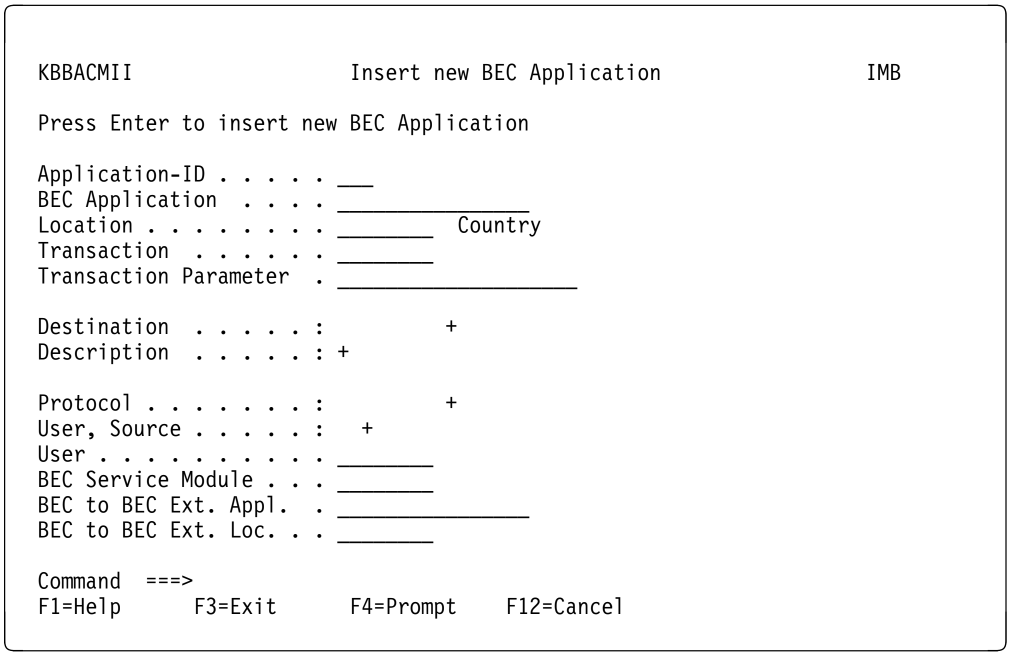
- Insert
- Copy
- Modify
- Delete
- Select

One example is the BEC Application Administration part of IMB from where all the actions can be given (see Figure 31 through Figure 36 on page 179).



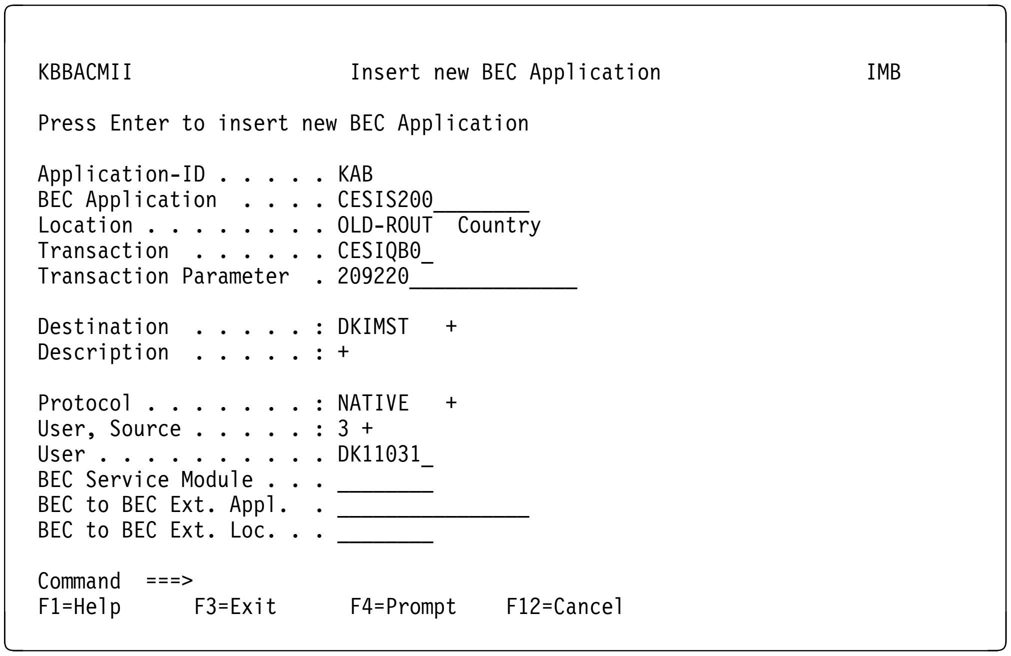
```
KBBABMIL                       BEC Applications                      IMB

Type one or more action codes, then press Enter.
Action codes: D=Delete  I=Insert  M=Modify  S=Select  C=Copy


   ____     ________________  ________  ________
A  Appl.    Name              Location  Transaction  Transaction Parm.
_  KAB      CESIS100          678       CESIQA0      209210
_  KAB      CESIS120          678       CESIQA1      209211
_  KAB      CESIS200          678       CESIQB0      209220
_  KAB      CESIS650          678       CESIQGI      209271
_  KAB      CESIS660          678       CESIQGK      209272
_  KAB      CESIS670          678       CESIQGM      209273
_  KAB      CESIS700          678       CESIQM0      209274
_  KAB      TEST              678       SS
_  KAE      FOS-MN-01         806       UPNOT0MN     4MN
_  KAE      FOS-MN-01         678       UPOST0MN     4MN
_  KAE      FOS-MN-01         846       UPSET0MN     4MN
_  KAE      FOS-MN-01         702       UPFIT0MN     4MN
_
Command  ===>
F1=Help      F3=Exit      F8=Forward   F12=Cancel
```

Figure 31. List panel



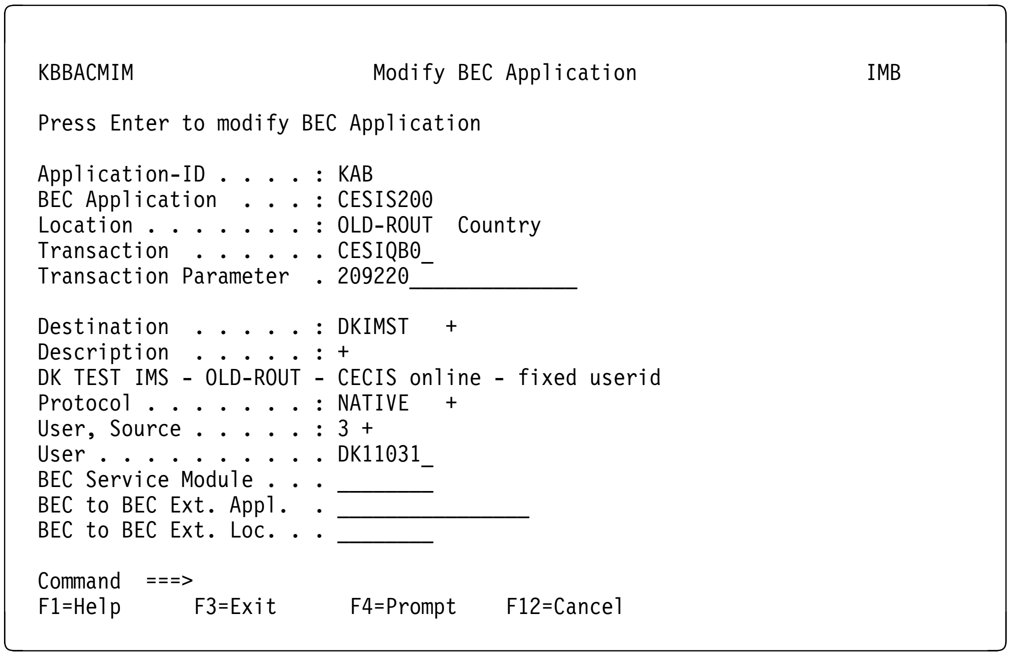
```
KBBACMII                  Insert new BEC Application                 IMB

Press Enter to insert new BEC Application

Application-ID . . . . . ___
BEC Application  . . . . ________________
Location . . . . . . . . ________  Country
Transaction  . . . . . . ________
Transaction Parameter  . ____________________

Destination  . . . . . :         +
Description  . . . . . : +

Protocol . . . . . . . :         +
User, Source . . . . . :   +
User . . . . . . . . . . ________
BEC Service Module . . . ________
BEC to BEC Ext. Appl.  . ________________
BEC to BEC Ext. Loc. . . ________

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

Figure 32. Detail panel (Insert)



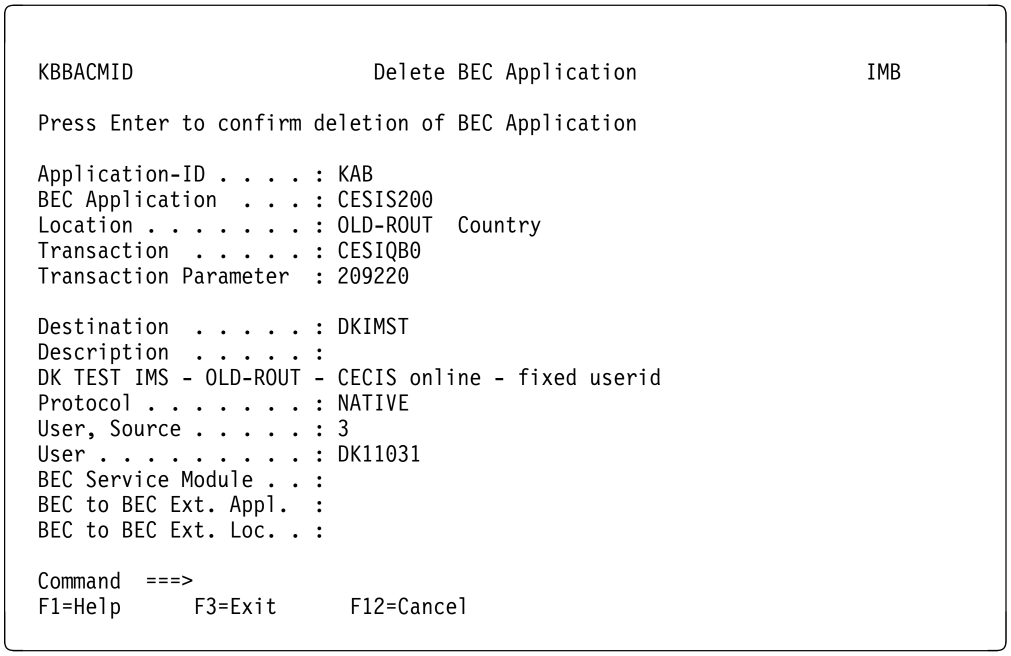
```
KBBACMII                  Insert new BEC Application                 IMB

Press Enter to insert new BEC Application

Application-ID . . . . . KAB
BEC Application  . . . . CESIS200________
Location . . . . . . . . OLD-ROUT  Country
Transaction  . . . . . . CESIQB0_
Transaction Parameter  . 209220______________

Destination  . . . . . : DKIMST   +
Description  . . . . . : +

Protocol . . . . . . . : NATIVE   +
User, Source . . . . . : 3 +
User . . . . . . . . . . DK11031_
BEC Service Module . . . ________
BEC to BEC Ext. Appl.  . ________________
BEC to BEC Ext. Loc. . . ________

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

Figure 33. Detail panel (Action)



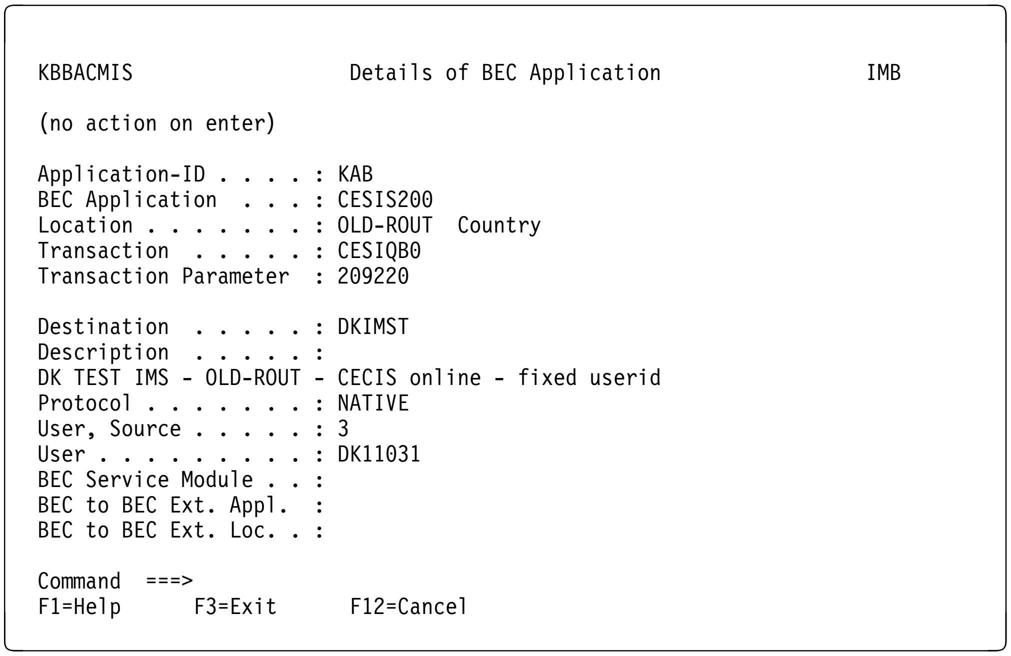
```
KBBACMIM                    Modify BEC Application                   IMB

Press Enter to modify BEC Application

Application-ID . . . . : KAB
BEC Application  . . . : CESIS200
Location . . . . . . . : OLD-ROUT  Country
Transaction  . . . . . . CESIQB0_
Transaction Parameter  . 209220______________

Destination  . . . . . : DKIMST   +
Description  . . . . . : +
DK TEST IMS - OLD-ROUT - CECIS online - fixed userid
Protocol . . . . . . . : NATIVE   +
User, Source . . . . . : 3 +
User . . . . . . . . . . DK11031_
BEC Service Module . . . ________
BEC to BEC Ext. Appl.  . ________________
BEC to BEC Ext. Loc. . . ________

Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

Figure 34. Detail panel (Action)



```
KBBACMID                    Delete BEC Application                   IMB

Press Enter to confirm deletion of BEC Application

Application-ID . . . . : KAB
BEC Application  . . . : CESIS200
Location . . . . . . . : OLD-ROUT  Country
Transaction  . . . . . : CESIQB0
Transaction Parameter  : 209220

Destination  . . . . . : DKIMST
Description  . . . . . :
DK TEST IMS - OLD-ROUT - CECIS online - fixed userid
Protocol . . . . . . . : NATIVE
User, Source . . . . . : 3
User . . . . . . . . . : DK11031
BEC Service Module . . :
BEC to BEC Ext. Appl.  :
BEC to BEC Ext. Loc. . :

Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 35. Detail panel (Action)



```
KBBACMIS                Details of BEC Application                IMB

(no action on enter)

Application-ID . . . . : KAB
BEC Application  . . . : CESIS200
Location . . . . . . . : OLD-ROUT  Country
Transaction  . . . . . : CESIQB0
Transaction Parameter  : 209220

Destination  . . . . . : DKIMST
Description  . . . . . :
DK TEST IMS - OLD-ROUT - CECIS online - fixed userid
Protocol . . . . . . . : NATIVE
User, Source . . . . . : 3
User . . . . . . . . . : DK11031
BEC Service Module . . :
BEC to BEC Ext. Appl.  :
BEC to BEC Ext. Loc. . :

Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 36. Detail panel (Action)

Multi Language Support (MLS)

IMB has implemented Multi Language Support.

A Detail application tries to find the texts for the panel in the language preferred by the user (ZCUSRLAN in KAAWCOM). If this language is not supported English is used instead.

CSP Tables

A Detail application uses two CSP tables that must be changed by the developer.

sssbbTP - F-Key table
This table holds information of the F-Key text in the supported languages. Only the fields marked in UPDCOLS will be updated.

DEL - Delete a row.
The *old* key (and timestamp) is passed in sssnnI_DB2ROW(1).

If the timestamp (DSYSRPT) given from the "old" row is not equal to the timestamp found on the table it has been changed after the browser has read it, and the Updater will return with an error without deleting the row.

SEL - Return a row matching the key(s) given.
The key is passed in sssnnI_DB2ROW(1).

APPLNAM - Application Name
Name of calling application for error logging purposes.

USERID - User Id
Name of user to be registrated in the field IUSRUUL.

sssnnI_DB2ROW
Input DB2 Rows.

UPDCOLS
Mark for update of field.

Any serious SQL errors are logged.

The Updater returns the following in the output record sssnnWP6:

MSGNO - Message Number
A message code to an IMB message if something went wrong.

RETCODE - Return code
4 is a warning on errors e.g. row updated by another user and duplicate keys found.

8 is an error code on errors e.g. not correct input and DB2 errors.

SQLCODE
SQL Error Code.

TABLEID
DB2 Table ID.

REFNAME
Name of Referential Integrity constraint, if any problems with referential integrity.

An 8 character name of the Referential Integrity constraint.

sssnnI_DB2ROW
The returned row (Insert / Modify / Select)

Referential Integrity

If you have a parent-child relation between two tables like the User table and the User Scope table shown in Figure 37, these tables should be protected by a referential integrity relation. This prevents a parent with children to be deleted or children to be inserted without a parent.

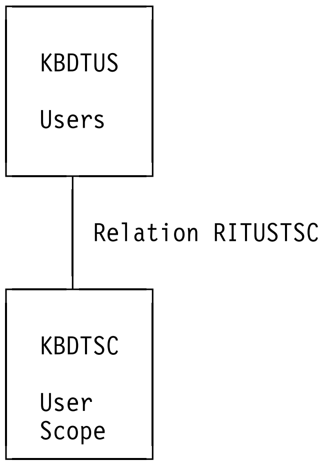


Figure 37. Parent-child referential relation

This relation is made in DB2 and is given an 8 character name. We recommend the following format: RITppTcc, where pp is a two char id of the parent table and cc is a two char id of the child table. The relation RITUSTSC is read like this :

Referential Integrity relation between parent Table (KBDT)US and child Table (KBDT)SC.

If the Updater tries to violate a referential relation, DB2 returns EZESQCOD = -530, -531, or -532 and the relation name in EZESQRRM.

In this cases the Updater tries to find a message in the csp table sssnnT1 using the relation name as key. If this does not succeed, a general message is used instead.

To show a more user-friendly message do like this:

1. Find the relation name.
2. Insert a line in sssnnT1 where:

   REFNAME Relation name.

   MSGNO Identifier of the IMB message to be shown.

   TEXT Description (not used).
3. Define the message in the IMB message table.
4. Load the message table.

An Update application can be generated automatically with the IMB Development skeletons (only available on request). The generated code will need only minor SQL modification.

Chapter 15. Back-end programming

All business data to be accessed from IMB has to be placed in back-end systems, preventing external users to have direct access to these data.

IMB Front-End applications in consequence must distribute part of the application to a back-end.

The objective of BEC is to pass data and control from a CICS Front-End to an IMS- or CICS back-end and to receive data and control again, once the back-end processing is complete.

This chapter will give design and development guidelines on how to use BEC when communicating between a BPI module (or NPT Front-End) and an application server in CICS or IMS.

Calling BEC

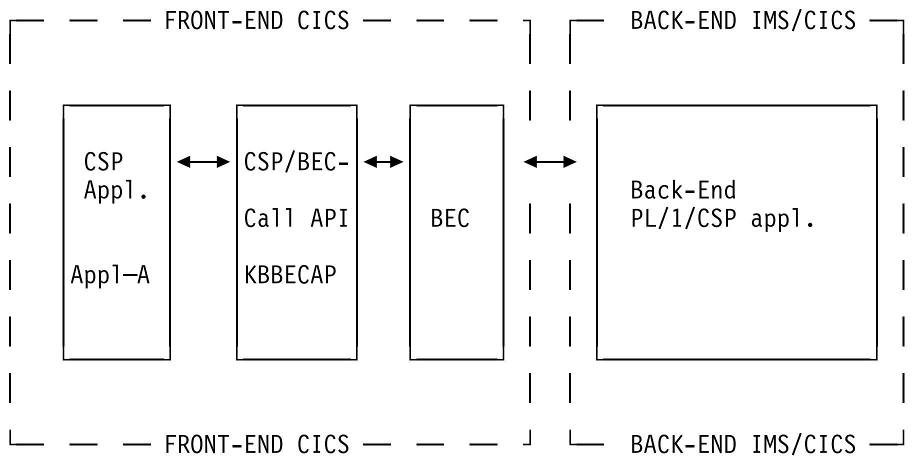


Figure 38. Front-End/back-end communication through BEC.

Figure 38 shows a schematic outline of the communication between a Front-End- and a back-end system. The **Front-End** is the CICS where IMB is installed.

The business logic of the application is placed in the **back-end**. All data access is performed in the back-end

CSP application **Appl_A** represents an NPT or BPI application installed on IMB.

Between the Front-End and the back-end is the **BEC** component, that handles the actual call to the back-end system. The Front-End applications do not need to know the physical location of the back-end; when a Front-End application calls BEC to access data from the back-end application, only a **logical destination** address is passed. The **physical destination** (the VTAM node for the back-end, the back-end transaction name, etc.) is then found by BEC from the IMB BEC tables. The on-line IMB administration utilities provide facilities for the definition and maintenance of logical and physical destinations and their connection. The **logical destination** is defined by two fields:

1. APPLCODE
2. LOCATION

The **APPLCODE** and **LOCATION** together make up the **logical destination** which is a unique identification of a BEC registration, i.e. an identification of the backend system to communicate with. **APPLCODE** is the only element that has to be "hardcoded" in the Client application/BPI definition. **LOCATION** enables execution in MCO mode. The **LOCATION** parameter must be filled in with a logical location code (usually the user's country code).

The BEC registration is performed from the online administration dialogue in IMB.

The Front-End application **Appl_A** communicates with BEC through the. **CSP/BEC call API**

The **CSP/BEC call API** is an IMB infrastructure component that handles the call to BEC. The call-interface to the CSP/BEC call API is described in details in the skeleton sample application **BECAPI** which can be found in the skeleton MSL, provided with the IMB package. The skeleton contains all the necessary code and parameters and can be copied directly into the application (as a process) or used as skeleton for a called CSP application to be used as a general component in your application complex. See also figure Figure 42 on page 187

The call statement for calling KBBECAP is:

Call Syntax (CSP)

```
►►──CALL KBBECAP KBBECWA,KBBECW_SEND,KBBECW_RECV (NOMAPS;──────────►◄
```

KBBECWA

The structure of the parameter KBBECWA is shown in Figure 39.

```
    NAME                 LEVEL OCCURS TYPE LENGTH   DESCRIPTION
***
001 IUSRIDY               10   00001  CHA  00008    IMB user ID
002 ICUSPRM               10   00001  CHA  00009    Primary Customer no.
003 *                     20   00001  CHA  00002
004 ICUSPRM_BEC           20   00001  CHA  00007    Primary cust. no. (BEC)
005 *                     10   00001  CHA  00003
006 LOCATION              10   00001  CHA  00008    Location / Country code
007 APPLCODE              10   00001  CHA  00016    Application code
008 MSGNO                 10   00001  CHA  00006    Message number
009 MSG                   10   00001  CHA  00071    Message text
***
```

Figure 39. CSP working storage KBBECWA. KBBECWA parameter to KBBECAP

IUSRIDY

Optional. You can provide the back-end application with a user ID in 3 different ways:

1. With the user ID signed on

2. With a user ID passed in the KBBECWA parameter (filled in by the application)
3. With a fixed user ID registered with the BEC application registration.

KBBECWA.IUSRIDY
This field must be completed before calling the BEC, if *User source* has been registered with a value of 2 (user ID to be passed in the KBBECWA parameter) in the BEC application registration online dialogue.

ICUSPRM
Optional. The primary customer number of the user.. The primary customer number is in the common communication working storage KAAWCOM.

ICUSPRM_BEC
(Used by BEC). Substructure to identify IBM account number.

LOCATION
Mandatory. The logical location of the back-end (typically country code of the user). The country code is in the common communication working storage KAAWCOM.

APPLCODE
Mandatory. The hard-coded logical key to the BEC application registration.

MSGNO
Returned parameter from BEC. BEC returns a message number in this field after the call.

MSG Returned parameter from BEC. BEC returns a message text in this field after the call.

The front-end application must move **MSG** and **MSGNO** to **ZMSG** and **ZMSGNO** of **KAAWCOM** See also Figure 42 on page 187

KBBECW_SEND

The structure of the parameter KBBECW_SEND is shown in Figure 40.

```
    NAME                  LEVEL OCCURS TYPE LENGTH DESCRIPTION
***
001 KBBECI_SENDAREA        05   00001  CHA  31885  Send area
002 QDEXLEN                10   00001  BIN  00004  length of interface
003 KBBECI_SENDDATA        10   00001  CHA  31883  Send area data
***
```

Figure 40. CSP working storage KBBECW_SEND. KBBECW_SEND parameter to KBBECAP

The working storage KBBECW_SEND.KBBECI_SENDDATA is filled in with application data to be passed from the calling Front-End application to the back-end application.

The field **QDEXLEN** specifies the length of the data to be passed to the back-end. The maximum length is 31883 bytes. The calling application must make a redefinition of the KBBECW_SEND area, which must match the redefinition in the corresponding back-end module.

KBBECW_RECV

The structure of the CSP working storage KBBECW_RECV is shown in Figure 41.

```
    NAME                                LEVEL OCCURS TYPE LENGTH DEC BYTES
***
001 KBBECI_RECVAREA                      05   00001  CHA  31885      31885
002 QDEXLEN                              10   00001  BIN  00004      00002
003 KBBECI_RECVDATA                      10   00001  CHA  31883      31883
```

Figure 41. CSP working storage KBBECW_RECV

The working storage KBBECW_RECV.KBBECI_RECVDATA is used to pass application data back from the back-end application to the calling Front-End application.

The field **QDEXLEN** specifies the length of the data passed back to the Front-End. The maximum length is 31885 bytes. The calling application must make a redefinition of the KBBECW_RECV area, which must match the redefinition in the corresponding back-end module.

For each call of the back-end via BEC, the working storage KBBECW_RECV must be appropriately redefined to match the corresponding redefinition of the WS-area in the back-end application. Figure 42 on page 187 shows how the CSP/BEC call API (and BEC) is called, and how messages from BEC should be treated by the calling application, It is a piece of code taken from the application BECAPI located in the skeleton MSL, which is a part of the IMB package.

```
/**************************************************
/* Set parameters for BEC call (from KAAWCOM)
/**************************************************
/*
/**************************************************
/* Initiate send and receive areas (p5 & p6)
/**************************************************
/*
SET KBBECWA EMPTY;
SET SEND_REDEF_EXAMPLE EMPTY;    /* Redifinition of KBBECW_SEND
SET RECV_REDEF_EXAMPLE EMPTY;    /* Redifinition of KBBECW_RECV
/*
/**************************************************
/* Set necessary parameters form BEC call
/**************************************************
/*
MOVE 'XXXXXXXXXXXXXXXX' TO KBBECWA.APPLCODE;  /* IE: "ECINFO"
MOVE 'NNN' TO KBBECWA.LOCATION;               /* COUNTRY CODE
MOVE 'YYYYYYYY' TO KBBECWA.IUSRIDY;           /* user ID
MOVE 'IIIIIIII' TO KBBECWA.ICUSPRM;           /* Prim. custno
/*
/**************************************************
/* Set length of SEND and RECIEVE areas
/**************************************************
/*
MOVE N TO SEND_REDEF_EXAMPLE.QDEXLEN;/* max 31885 (length of interface)
MOVE I TO RECV_REDEF_EXAMPLE.QDEXLEN;/* max 31885 (length of interface)
/*
/**************************************************
/* Build KBBECW_SEND area with application data
/**************************************************
/*
MOVE XXX TO SEND_REDEF_EXAMPLE.XXX;
MOVE YYY TO SEND_REDEF_EXAMPLE.YYY;
/*
/**************************************************
/* CALL BEC CSP Call API
/**************************************************
/*
CALL KBBECAP
  KBBECWA,
  KBBECW_SEND,
  KBBECW_RECV
  (NOMAPS;
/*
/**************************************************
/* Return  error message and errortext
/**************************************************
/*
IF KBBECWA.MSGNO ¬= ' ';
  MOVE KBBECWA.MSGNO TO ZMSGNO;
  MOVE KBBECWA.MSG TO ZMSG;
END;
;
/*  proceed with application logic
```

Figure 42. Call CSP/BEC call API. Example on how to call the CSP/BEC call API.

CICS to CICS programming guidelines

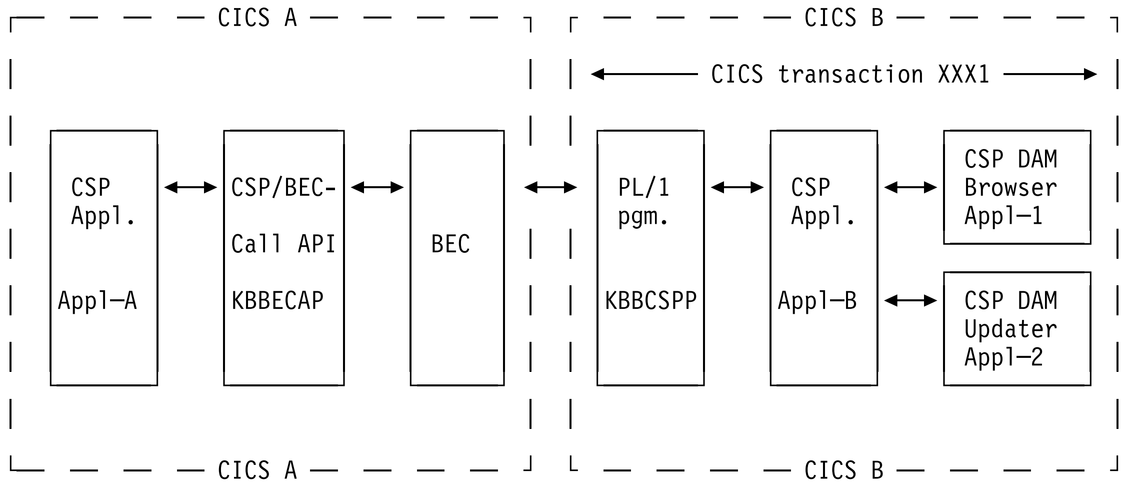


Figure 43. CICS to CICS..

Figure 43 shows a schematic outline of how an application is split between 2 CICS to separate business logic from the presentation logic, and distribute it to a back-end CICS.

Compared to Figure 38 on page 183 the PL/1 program **KBBCSPP** has been added. KBBCSPP is an IMB infrastructure component, which makes it possible to start a CSP application in the back-end CICS.

We will assume the following scenario:

We will install a "stub" application (**Appl_B**) in the back-end system to manage the request from the Front-End application. The Front-End application can request either a call to the browser module Appl_1 or the update module Appl_2.

This application must be a called CSP application with 1 input record and 1 output record.

It is also possible to install real business applications (i.e. a data access module) instead of a stub application. But by installing a stub application we minimize the administrative work and maintenance (e.g. CICS definitions.)

CSP Front-End application **Appl_A** gets data from the CSP access module **Appl_1**. It updates a DB2 table through CSP update module **Appl_2**. Both access modules must be placed in a CICS back-end system.

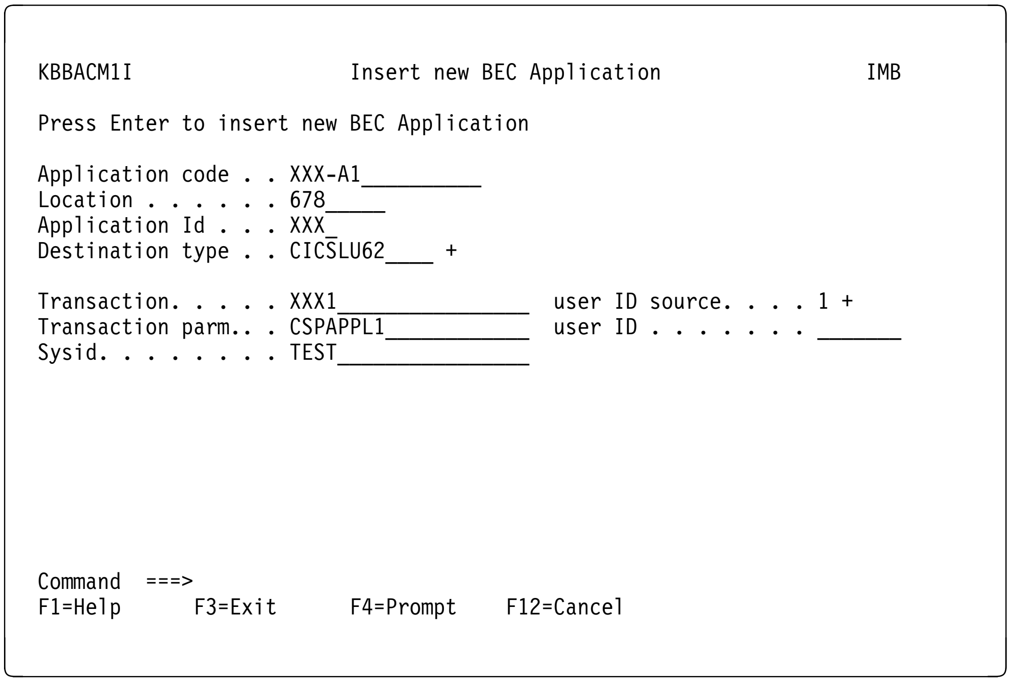
We will need a CICS transaction with the appropriate definitions in the **CICS RCT table**

We want our back-end applications to run in **CICS transaction "XXX1"**, and to execute through a **DB2 plan "XXXPLAN"**(static DB2).

We have been given**application id "XXX"** and we will use **Applcode "XXX-A1"**. Our back-end system is located in Denmark **Location "678"**

IMB BEC application registration

Assuming that the appropriate BEC destination has been defined, the BEC application definition panel in the IMB Application Installation should be filled in with the following parameters (refer to *System Administration Guide* for details about inserting BEC applications and destinations):



```
KBBACM1I                 Insert new BEC Application                IMB

Press Enter to insert new BEC Application

Application code . . XXX-A1__________
Location . . . . . . 678_____
Application Id . . . XXX_
Destination type . . CICSLU62____ +

Transaction. . . . . XXX1________________  user ID source. . . . 1 +
Transaction parm.. . CSPAPPL1____________  user ID . . . . . . . _______
Sysid. . . . . . . . TEST________________






Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

CEDA definition of transaction

A transaction XXX1 must be defined using CEDA. It should start program KBBLNKP, which then will link to the program from “Transaction parm,” here CSPAPPL1. The TWASIZE should also be changed to 1024 to allow the CSP program to run.

CICS RCT table

The CICS RCT (Resource control table) of the back-end CICS must have the following contents:

AUTH CICS authorization ID. This is the identifier that must be granted access to the DB2 resources.

TXID CICS transaction ID.

PLAN DB2 plan related to the CICS transaction. The AUTD ID (XYZ) must be granted execute to the plan (XXXPLAN).

```
*-------------------------------------------------------------------*
* CICS RCT TABLE FOR PROJECT ABC                                    *
*-------------------------------------------------------------------*
*
* The following RCT entries are to be applied to any CICS region
* where project ABC will execute.
*
*-------------------------------------------------------------------*
         DSNCRCT TYPE=ENTRY,                                         X
               TWAIT=POOL,THRDM=0,THRDA=0,AUTH=(XYZ),                X
               TXID=(XXX1),                                          X
               PLAN=XXXPLAN
```

CICS to IMS programming guidelines

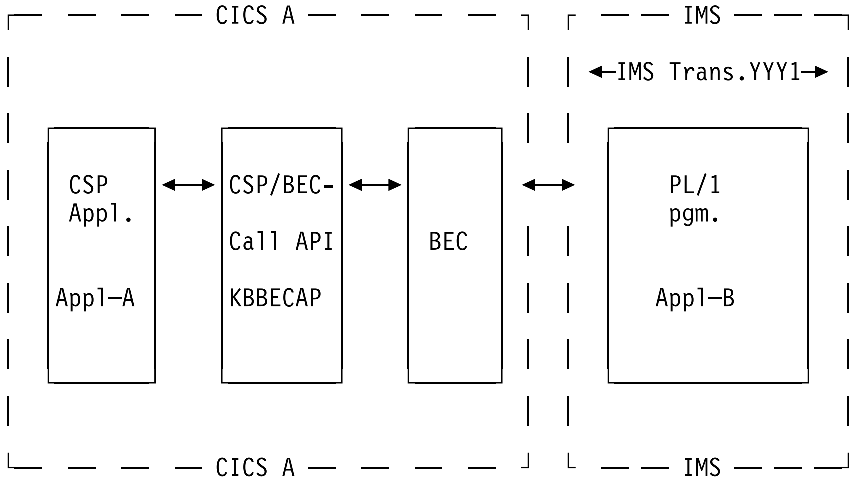


Figure 44. CICS to IMS..

Figure 44 shows a schematic outline of how an application is split between a CICS and a IMS in order to separate business logic from the presentation logic, and distribute it to a back-end IMS.

When BEC is called from the CSP/BEC call-API (**KBBCSPAP**), BEC will place the enriched **send_area** on the IMS message-queue. Figure 45 shows the structure of the send_area

```
Prefix      14 bytes  (Length of area and IMS transaction, filled in by BEC)
P3_area    115 bytes  (Protocol area, filled in by BEC)
P5_area  31883 bytes  (Application data input area)
```

Figure 45. IMS input message area

Figure 46 on page 191 shows the declare structure of the P3_area.

```
/*   +----------------------------------------------------------+    *
/*   |              BACKEND COMMUNACATIONS (BEC)                |    *
/*   |              ============================                |    *
/*   |                                                          |    *
/*   |  STRUCTURE : KBBI1PRO                                    |    *
/*   |                                                          |    *
/*   |  LENGTH    : 115 BYTES (DATA = 031, FUTURE USE = 084)    |    *
/*   |                                                          |    *
/*   |  CONTENTS  : IMB NATIVE APPLICATION PROTOCOL PREFIX      |    *
/*   |              SENT TO BACK-END.                           |    *
/*   |                                                          |    *
/*   |  RELATIONS : BUILT BY BEC PROTOCOL MODULE KBBCP1M AND    |    *
/*   |              RECOGNIZED BY CORRESPONDING SERVERS IN      |    *
/*   |              BACK-ENDS.                                  |    *
/*   |              HISTORICALLY THIS PARAMETER IS KNOWN AS     |    *
/*   |              ROUTER PARAMETER P3 MAPPED BY CSP WORKING   |    *
/*   |              STORAGE RECORD KAAWIP3.                     |    *
/*   |                                                          |    *
/*   +----------------------------------------------------------+    *
/*                                                                   *
/*                                                            OFFSET *
/*                                                               --- *
    3 LL         BIN FIXED (15),   /* RECORD LENGTH              000 *
    3 ZZ         BIN FIXED (15),   /* INTERFACE ID               002 *
    3 user ID      CHAR (08),        /* IMS USER ID                004 *
    3 FILLER1    CHAR (05),        /* FUTURE USE                 012 *
    3 SCOPEKEY   CHAR (07),        /* DATA ACCESS SCOPE KEY      017 *
    3 CHOICE     CHAR (12),        /* COMSEC TRANS.              024 *
    3 FILLER2    CHAR (79)         /* FUTURE USE                 036 *
/*                                                                   *
/*                                                  TOTAL LENGTH 115 *
/*  ==  BEC  == END OF STRUCTURE KBBI1PRO ==                         *
/*  ---------------------------------------------------------------- *
```

Figure 46. P3 area

When BEC has placed the send_area on the IMS message queue, the program related to the IMS transaction will be executed. The main structure of this program can be seen in Figure 47

Note that a call to **COMSEC** is optional. If COMSEC is used, you specify the **COMSEC choice number** in the transaction parameter field, when you define your BEC application from the IMB online dialog.(see below).

```
READ MESSAGE-QUEUE
DO WHILE MESSAGE FOUND
  FIND USER SCOPE FROM COMSEC     /* OPTIONAL  */
  CALL APPROPRIATE MODULE
  BUILD AND SEND RECEIVE_AREA
  READ MESSAGE-QUEUE
```

Figure 47. Main structure of main program on IMS site.

Figure 48 on page 192 shows the structure of the receive_area.

```
Prefix       6 bytes  (Length of area, filled in by BEC)
P4_area     21 bytes  (Protocol area, filled in by BEC)
P6_area  31883 bytes  (application data output area)
```

Figure 48. IMS reply message area

Figure 49 shows the declare structure of the p4_area.

```
/*   +----------------------------------------------------------+    *
/*   |              BACKEND COMMUNACATIONS (BEC)                |    *
/*   |              ============================                |    *
/*   |                                                          |    *
/*   |  STRUCTURE : KBBO1PRO                                    |    *
/*   |                                                          |    *
/*   |  LENGTH    : 021 BYTES                                   |    *
/*   |                                                          |    *
/*   |  CONTENTS  : IMB NATIVE APPLICATION PROTOCOL PREFIX      |    *
/*   |              RECEIVED FROM BACK-END                      |    *
/*   |                                                          |    *
/*   |  RELATIONS : BUILT BY BACK-END SERVER ON REPLY AND       |    *
/*   |              RECOGNIZED BY CORRESPONDING BEC PROTOCOL    |    *
/*   |              MODULE.                                     |    *
/*   |              HISTORICALLY THIS PARAMETER IS KNOWN AS     |    *
/*   |              ROUTER PARAMETER P4 MAPPED BY CSP WORKING   |    *
/*   |              STORAGE RECORD KAAWIP4.                     |    *
/*   |                                                          |    *
/*   +----------------------------------------------------------+    *
/*                                                                   *
/*                                                            OFFSET *
/*                                                               --- *
    3 LL         BIN FIXED (15),   /* RECORD LENGTH            000 *
    3 ZZ         BIN FIXED (15),   /* INTERFACE ID             002 *
    3 TRANCODE   CHAR (08),        /* IMS TRAN CODE            004 *
    3 ENVSTAT    CHAR (02),        /* ENV. STATUS CODE         012 *
    3 ENVMSG,                      /* ENV. MESSAGE NUMBER          *
      4 ENVMSGPR CHAR(3),          /*   PREFIX (E.G. 'KBB')    014 *
      4 ENVMSGNO CHAR(3),          /*   MSG NUMBER             017 *
      4 ENVMSGLV CHAR(1)           /*   MSG LEVEL              020 *
/*                                                                   *
/*                                                  TOTAL LENGTH 021 *
/*  ==  BEC  == END OF STRUCTURE KBBO1PRO ==                         *
/*  ---------------------------------------------------------------- *
```

Figure 49. P4 area

We will assume the following scenario:

We will install an application (**Appl_B**) in the back-end IMS system.

CSP Front-End application **Appl_A** gets data from the PL/1 program **Appl_B**.

We will need to supply the IMS generation with appropriate definitions.

We will use COMSEC and will use **COMSEC choice number: AAYY**

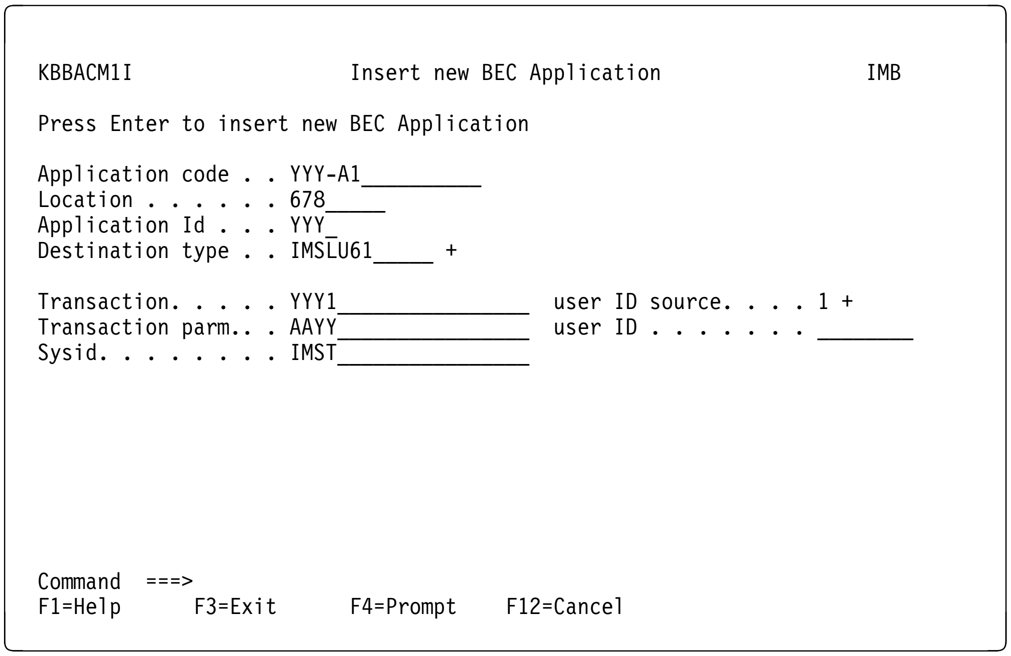
We want our back-end applications to run in **IMS transaction YYY1**

We want to use the **IMS NATIVE** protocol. (The IMS ADF protocol is described later in this chapter)

We have been given **application id YYY** and we will use **Applcode YYY-A1**. Our back-end system is located in Denmark **Location "678"**

IMB BEC application registration

Assuming that the appropriate BEC destination has been defined, the BEC application definition panel in the IMB Application Installation should be filled in with the following parameters: (refer to *System Administration Guide* for details about inserting BEC applications and destinations.).



```
KBBACM1I                 Insert new BEC Application                IMB

Press Enter to insert new BEC Application

Application code . . YYY-A1__________
Location . . . . . . 678_____
Application Id . . . YYY_
Destination type . . IMSLU61_____ +

Transaction. . . . . YYY1________________  user ID source. . . . 1 +
Transaction parm.. . AAYY________________  user ID . . . . . . . ________
Sysid. . . . . . . . IMST________________






Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

IMS Definitions

The Input data to be used for the IMS generation must have the following contents:

- PSB = name of PSB and PROGRAM to be started
- CODE = name of IMS transaction.

```
***********************************************************************
*
*   TRANSACTIONS IN IMS
*
***********************************************************************
*   NON CONVERSIONAL TRANSACTIONS FOR BACKEND MODULES
***********************************************************************
*
         APPLCTN PSB=YYY1P,PGMTYPE=(TP,,12),SCHDTYP=PARALLEL
         TRANSACT PRTY=(1,1,05),EDIT=(ULC),                            X
         TRANSACT PRTY=(1,1,05),EDIT=(ULC),                            X
               CODE=YYY1,MSGTYPE=(SNGLSEG,RESPONSE),PARLIM=1
```

Part 4. Common programming APIs

This section describes the application programming interfaces provided with IMB:

- Chapter 16, “Generic IMB programming APIs” on page 197 describes the generic IMB programming APIs
- Chapter 17, “Programming APIs and structures for CSP 3270 applications” on page 251 describes the APIs and structures for CSP 3270 applications

Chapter 16. Generic IMB programming APIs

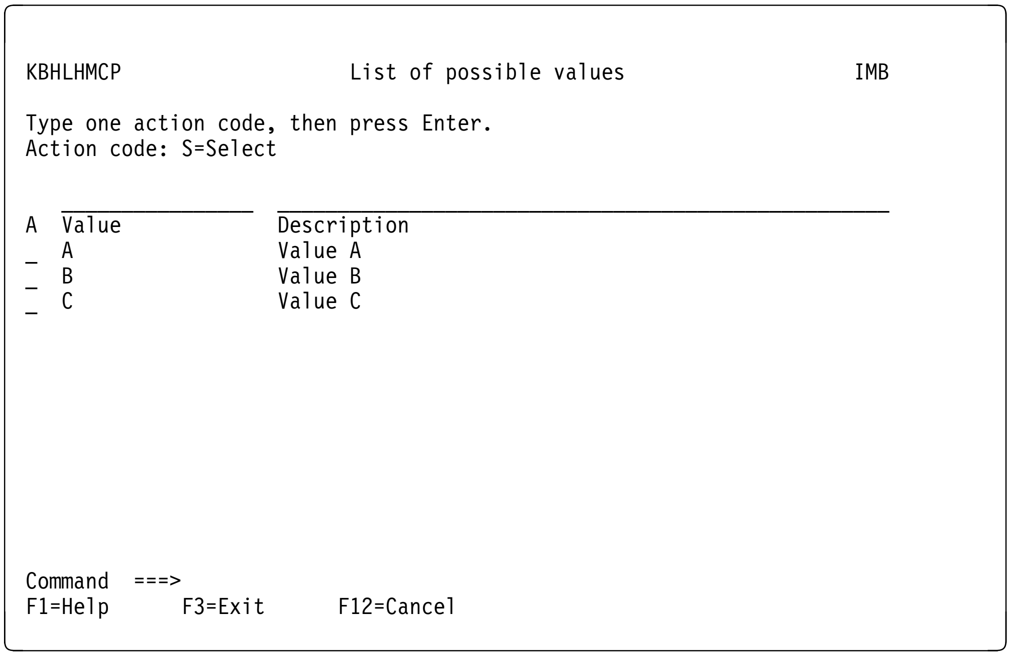
The following APIs can be used from most programs running in IMB CICS. Both online 3270/NPT applications written in CSP, and BPIs and MailRoom programs written in PL/1 can use IMB programming APIs.

Sundry texts, prompt and validation

The IMB Sundry text table is a generic place to store various character, decimal and integer values and related descriptions for later use in applications as F4 Prompt lists and input validation. It consists of a DB2 table to store the sundry texts, a CSP Call API to give a Prompt list, a CSP Call API to verify a value, and a set of CSP applications to maintain Sundry texts online under IMB.

Usage

The Sundry Prompt List is activated under program control, and can display a list of values like those shown in Figure 50).



```
KBHLHMCP                  List of possible values                 IMB

Type one action code, then press Enter.
Action code: S=Select

   ________________  ____________________________________________________
A  Value             Description
_  A                 Value A
_  B                 Value B
_  C                 Value C








Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 50. Sundry Prompt list

By selecting a line, the value (and description) is returned to the calling application for further usage.

Context-sensitive help can be available, if it has been written and loaded as described in “IMB online help system” on page 251. The used help key should match the sundry key.

Call Method

The Sundry Text APIs can be called in a number of different ways, depending on which support is needed.

- Online Prompt List
- Return a list of values to an application
- Check a character value against the list
- Check an integer value against the list
- Check a decimal value against the list
- Return a list of values to an application where the value field is part of the search parameter.

Sundry texts are organized in groups of Sundry Items, which can have one or more values (in either character, integer or decimal representation). The identification of a Sundry Item is:

- Application Id (char 4)
- Country code (char 3)
- Language code (char 2)
- Sundry Item (char 8)
- Value (char 16, optional)

To support MCO and NLS, the APIs are able to default to another language code or country, if a Sundry Item is not registered.

The following examples show how the Sundry texts can be accessed from CSP.

```
/******************************************/
/* Call of Sundry Text Prompt List        */
/* to display values for a field          */
/* CSP procedure: KBHxxP_PROMPT           */
/******************************************/
;
MOVEA ' ' TO KBHxxWM.CURSOR;
;
SET KBHLHWA EMPTY;
MOVE 'ALL' TO KBHLHWA.WHO;
MOVE 'PRO' TO KBHLHWA.WHAT;
MOVE 'Y' TO KBHLHWA.ENABLE_FLD_HELP;
MOVE 'KBH' TO KBHLHWA.ISYSIDY;
MOVE ZIOPUCTY TO KBHLHWA.IOPUCTY;
MOVE ZCUSRLAN TO KBHLHWA.CUSRLAN;
MOVE 'PROMPT-1' TO KBHLHWA.SKBH_SUNITEM;
; /* ********************************************
; /* The default sort in the Sundry Text Prompt List
; /* is sort by Sundry Value.
; /* If the Sundry Text Prompt List should be sorted
; /* by Sundry Text instead, then remove this comment
; MOVE 'T' TO KBHLHWA.KBHLHI_SKBH_SORT;
;
CALL KBHLHAP KBHLHWA,KAAWCOM;
IF ZNEWAPPL ¬= ' ';
  EZECLOS;
END;
;
IF ZMSGNO = ' '                  /* No errors
  AND KBHLHWA.CANCEL = 'NO';     /* Enter pressed
  ;/* OK user selected a value
  MOVE MOVE KBHLHWA.SKBH_SUNVALUE TO KBHxxWM.xxxxxxxx;
  MOVE MOVE KBHLHWA.SKBH_SUNTEXT TO KBHxxWM.xxxxxyyy;
  MOVE 'YES' TO CURSOR(2);       /* place cursor here
END;
;
```

Figure 51. Sundry Text Prompt List

```
/*****************************************/
/* Call of Sundry Text Pmg API           */
/* to list all values                    */
/*****************************************/
MOVE ZUSERID TO KBHLGWA.USERID;
MOVE 'LST' TO KBHLGWA.FUNC;
MOVE 'KBH' TO KBHLGWA.ISYSIDY;
MOVE ZIOPUCTY TO KBHLGWA.IOPUCTY;
MOVE ZCUSRLAN TO KBHLGWA.CUSRLAN;
MOVE 'PROMPT-1' TO KBHLGWA.SKBH_SUNITEM;
;
CALL KBHLGAP KBHLGWA (NOMAPS;
;
IF KBHLGWA.MSGNO = ' ';    /* found
  ; /*
  ; /* The list is available in
  ; /*  KBHLGWA.SKBH_SUNVALUE      (Char type)
  ; /*  KBHLGWA.SKBH_SUNVAL_I      (Integer type)
  ; /*  KBHLGWA.SKBH_SUNVAL_D      (Decimal type)
  ; /*  KBHLGWA.SKBH_SUNTEXT       (Text description)
  ; /*
  ; /* Info about number of rows
  ; /*  KBHLGWA.QROWCNT
  ; /*  KBHLGWA.MORE
  ; /*
  ; /* Info about actually used cty and lang
  ; /*  KBHLGWA.IOPUCTY
  ; /*  KBHLGWA.CUSRLAN
  ; /*
ELSE;
  MOVE KBHLGWA.MSGNO TO ZMSGNO;
END;
```

Figure 52. Sundry, list valid values

```
/******************************************/
/* Call of Sundry Text Pmg API            */
/* to verify a value                      */
/******************************************/
MOVE ZUSERID TO KBHLGWA.USERID;
MOVE 'CHK' TO KBHLGWA.FUNC;
; /* 'CHI' for Integer values
; /* 'CHD' for decimal values
MOVE 'KBH' TO KBHLGWA.ISYSIDY;
MOVE ZIOPUCTY TO KBHLGWA.IOPUCTY;
MOVE ZCUSRLAN TO KBHLGWA.CUSRLAN;
MOVE 'PROMPT-1' TO KBHLGWA.SKBH_SUNITEM;
MOVE xxmyvalue TO KBHLGWA.SKBH_SUNVALUE;
; /* For integer values:  KBHLGWA.SKBH_SUNVAL_I
; /*     decimal values:  KBHLGWA.SKBH_SUNVAL_D
;
CALL KBHLGAP KBHLGWA (NOMAPS;
;
IF KBHLGWA.MSGNO = ' ';    /* found
  ; /*
  ; /* The list is available in
  ; /* (there is normally only one row...)
  ; /*  KBHLGWA.SKBH_SUNVALUE      (Char type)
  ; /*  KBHLGWA.SKBH_SUNVAL_I      (Integer type)
  ; /*  KBHLGWA.SKBH_SUNVAL_D      (Decimal type)
  ; /*  KBHLGWA.SKBH_SUNTEXT       (Text description)
  ; /*
  ; /* Info about number of rows
  ; /*  KBHLGWA.QROWCNT
  ; /*  KBHLGWA.MORE
  ; /*
  ; /* Info about actually used cty and lang
  ; /*  KBHLGWA.IOPUCTY
  ; /*  KBHLGWA.CUSRLAN
  ; /*
ELSE;
  IF KBHLGWA.MSGNO = 'KAA100';   /* not found
    ; /* Your value is wrong
  ELSE;
    MOVE KBHLGWA.MSGNO TO ZMSGNO;
  END;
END;
```

Figure 53. Sundry, verify a value

```
/*****************************************/
/* Call of Sundry Text Pmg API           */
/* to get a list where the input parameter */
/* Value contains the first part of the   */
/* word in the output parameter Value.    */
/* Like a *-search.                       */
/*****************************************/
MOVE ZUSERID TO KBHLGWA.USERID;
MOVE 'BCK' TO KBHLGWA.FUNC;
MOVE 'KBH' TO KBHLGWA.ISYSIDY;
MOVE ZIOPUCTY TO KBHLGWA.IOPUCTY;
MOVE ZCUSRLAN TO KBHLGWA.CUSRLAN;
MOVE 'PROMPT-1' TO KBHLGWA.SKBH_SUNITEM;
MOVE xxmyvalue TO KBHLGWA.SKBH_SUNVALUE;
;
CALL KBHLGAP KBHLGWA (NOMAPS;
;
IF KBHLGWA.MSGNO = ' ';    /* found
  ; /*
  ; /* The list is available in
  ; /*  KBHLGWA.SKBH_SUNVALUE      (Char type)
  ; /*  KBHLGWA.SKBH_SUNTEXT       (Text description)
  ; /*
  ; /* Info about number of rows
  ; /*  KBHLGWA.QROWCNT
  ; /*  KBHLGWA.MORE
  ; /*
  ; /* Info about actually used cty and lang
  ; /*  KBHLGWA.IOPUCTY
  ; /*  KBHLGWA.CUSRLAN
  ; /*
ELSE;
  IF KBHLGWA.MSGNO = 'KAA100';   /* not found
    ; /* Your value is wrong
  ELSE;
    MOVE KBHLGWA.MSGNO TO ZMSGNO;
  END;
END;
```

Figure 54. Sundry between character search. Return a list of values to an application where the value field is part of the search parameter.

System Errorlog

If an error occurs in an application program, information is often available regarding the problem. This information can be important for later problem resolution, but the application might not need or understand it. It is nevertheless important to store the information for later use. The IMB System Errorlog is used for this purpose.

The System Errorlog can be used for these types of errors:

SQL errors from DB2

The information returned in the SQL Communication Area is in packed format. This data is useful for the DBA or programmer during problem determination. The user needs a more basic presentation of the error information. Storing the information in a DB2 Table lets you keep the information for later tracing of the problem. The System Errorlog formats SQLCA into readable text by using the DB2 system routine DSNTIAR.

Simple errors with a message number
The application might have so important an error message that it must be kept and possibly automatically sent to someone in the form of alert.

The System Errorlog looks up the message number in the DB2 message table, substitutes any variables and formats the resulting text.

Text information
The application can use the API to store up to 7 lines of formatted text.

Combination of message number and free text
A combination of the previous two message types.

CICS errors and message number
Support for handling and formatting CICS errors. This is the support as for plain messages with the addition of automatic resolving of CICS EIBRESP and EIBRESP2 into the text.

IMS errors and message number
Support for formatting IMS errors. This is the support as for plain messages with the addition of extra information about the IMS system and Lterm where the error was encountered.

Using System Errorlog

The System Errorlog is available as an API for:

- PL/1 programs using an include member. Working in these environments by linking an environment-specific module into the main load module:
  - CICS
  - IMS
  - Batch TSO
- CICS programs using EXEC CICS LINK
- CSP applications (under CICS) through a called application

Processing in CICS environment

A CICS main program is the real API to System Errorlog. Other CICS programs and CSP applications can call it directly. PL/I programs will call (through an include member) a generic API. In the CICS environment the call is eventually transformed to a LINK to the real API by including the CICS version of the System Errorlog in the final load module.

The real API will initialize some parameters like time, user ID, CICS system, transaction, task, etc. then write a record to a TD Queue and return to the original program for further processing. The actual insertion into the System Errorlog will take place in a different task.

Processing in an IMS environment

PL/I programs will call (through an include member) a generic API. In the IMS environment the call results in the insertion of a message to the IMS queue, which starts an IMS transaction with the System Errorlog structure.

The IMS transaction performs the insertion into the System Errorlog. If the TIE-MQ scenario is being used, the IMS transaction formats the message for MQSeries and puts it on the queue for errors.

If there is a general DB2 error or a plan error on the IMS transaction for error handling, the transaction fails and the error message is written to the IMS region.

Processing in a batch TSO environment with DB2 access

PL/I programs will call (through an include member) a generic API. In the BATCH/TSO environment the call is eventually transformed to an insertion into the System Errorlog.

The insertion will fail if the current DB2 plan is not working correctly. In that case, the formatted error message will only be written to SYSPRINT.

Processing in a batch TSO environment without DB2 access

PL/I programs will call (through an include member) a generic API. In the BATCH/TSO environment the call is eventually transformed into writing a formatted error message to SYSPRINT.

Calling System Errorlog

The System Errorlog is called in 2 functionally different ways:

- Entry point with parameters from PL/I
- Link to API with structure from CICS and CSP

The required parameters for the entry points as well as the fields in the structures are shown in Table 13.

Table 13 (Page 1 of 2). System Errorlog. Parameters to the API

Parameter	parms for entry point fields in structure					Description
	MSG	DB2	CICS	GEN	GMS	
SKBH_ELOG_TYPE	1	2	3	4	5	Type of call
SKBH_ELOG_PROG	•	•	•	•	•	Calling program
SKBH_ELOG_SUBRUT	•	•	•	•	•	Process in program
SKBH_ELOG_CFROM	•	•	•	•	•	Called from
SKBH_ELOG_OBJECT	•	•	•	•	•	Related object
SKBH_ELOG_SUBSYS	•	•	•	•	•	Subsystem
SKBH_ELOG_ALERT	•	•	•	•	•	Alert Y/N
SKBH_ELOG_SYS	□	□	□	□	□	Assigned by API
SKBH_ELOG_TIME	□	□	□	□	□	Assigned by API
SKBH_ELOG_TRX	□	□	□	□	□	Assigned by API
SKBH_ELOG_TASK	□	□	□	□	□	Assigned by API
SKBH_ELOG_TERM	□	□	□	□	□	Assigned by API
SKBH_ELOG_USER	□	□	□	□	□	Assigned by API
SKBH_MSGID	•		•	•	•	Message id
SKBH_MSGSEVR				•		Message severity
SKBH_MSGVAR1	•		•		•	Message variable 1
SKBH_MSGVAR2	•		•		•	Message variable 2

Table 13 (Page 2 of 2). System Errorlog. Parameters to the API

Parameter	**parms for entry point fields in structure**					**Description**
	MSG	**DB2**	**CICS**	**GEN**	**GMS**	
SKBH_MSGVAR3	•		•		•	Message variable 3
SKBH_ELOG_EIBRESP			•			CICS EIBRESP
SKBH_ELOG_EIBRESP2			•			CICS EIBRESP2
SKBH_ELOG_TEXT1				•	•	Text line 1
SKBH_ELOG_TEXT2				•	•	Text line 2
SKBH_ELOG_TEXT3				•	•	Text line 3
SKBH_ELOG_TEXT4				•	•	Text line 4
SKBH_ELOG_TEXT5				•	•	Text line 5
SKBH_ELOG_TEXT6				•		Text line 6
SKBH_ELOG_TEXT7				•		Text line 7
SKBH_ELOG_SQLCA		•				DB2 SQLCA
RESERVED	□	□	□	□	□	Filler, leave blank

Legend: The symbols used in the table are interpreted as follows:

For PL/I entry points a • indicates a required parameter
MSG: Use the KBHLMSG entry point to store a message no
DB2: Use the KBHLDB2 entry point to store SQL errors from DB2
CICS: Use the KBHLCIC entry point to store CICS errors
GEN: Use the KBHLGEN entry point to store generic text
GMS: Use the KBHLGEN entry point to store message and generic text

For CICS/CSP API a •, □, 1-5 indicates field in structure
MSG: Use the KBHERWLM structure to store a message no
DB2: Use the KBHERWLD structure to store SQL errors from DB2
CICS: Use the KBHERWLC structure to store CICS errors
GEN: Use the KBHERWLG structure to store generic text
GMS: Use the KBHERWLJ structure to store message and generic text
•: indicates a field where input is required
□: indicates a field where input must be blank
Use value 1:'MSG', 2:'SQL', 3:'CICS', 4:'GEN', 5:'GMS'

Calling the API from PL/I

The PL/I program must include the KBHERROR member. This member will define 5 entry points to be used when calling the System Errorlog. The 5 entry points are resolved to real code by including the relevant environment version of the API in the linkage editor:

```
INCLUDE SYSNCAL(KBHERCM)     Errorlog module for CICS
INCLUDE SYSNCAL(KBHERIM)     Errorlog module for IMS
INCLUDE SYSNCAL(KBHERTM)     Errorlog module for TSO/BATCH with DB2
INCLUDE SYSNCAL(KBHERBM)     Errorlog module for TSO/BATCH without DB2
```

The API is exactly the same, but the internal processing will vary in the different environments.

The following is a sample of calling the System Errorlog from PL/I in order to store a CICS error:

```
%INCLUDE KBHERROR;
CALL KBHLCIC(MY_MODULE,           /* name of this module      */
             'XYZ_PROCESS',       /* name of process          */
             BEC_XXX.TOPPGM,      /* name of top lvl pgm      */
             BEC_COMM.APPLCODE,   /* name of current 'object' */
             'BEC',               /* name of subsystem        */
             'Y',                 /* Alert=Yes                */
             'MSG123',            /* message number           */
             COMM.LUNAME          /* variable 1               */
             'EIBRESP'            /* var 2, put CICS msg here */
             '',                  /* variable 3 not used      */
             EIBRESP,             /* EIBRESP  from CICS EIB   */
             EIBRESP2);           /* EIBRESP2 from CICS EIB   */
```

If MSG123 has a message text like `'Error accessing LU=&1 (tech. reason: &2)'` the resulting message in the System Errorlog could be something like: `'Error accessing LU=FKBZ1234 (tech. reason: TERMERR RESP2=4)'`

Calling the API from CSP

CSP is using the EXEC CICS LINK version of the API. The calling program must prepare a working storage area, and then call the CICS program. Other programs in CICS can use the same API (the CICS commarea must contain a pointer to the structure).

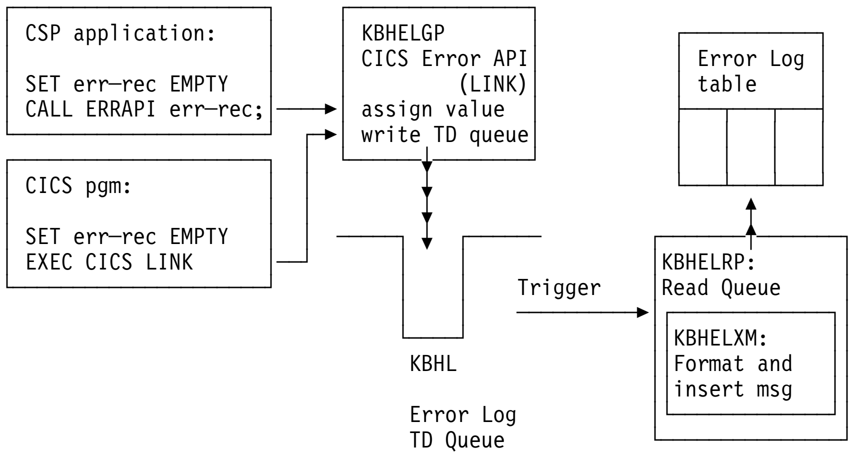
The following is a sample of calling the System Errorlog from CSP in order to store a message number with variables:

```
Include the following records in the application (option 3)
(you may leave out any unused redefinitions)
     KBHERWL
     KBHERWLC          (redefinition of KBHERWL for CICS errors)
     KBHERWLD          (    -"-          -"-   for DB2  errors)
     KBHERWLG          (    -"-          -"-   for generic text)
     KBHERWLJ          (    -"-          -"-   for msg + gen. text)
     KBHERWLM          (    -"-          -"-   for plain message no)

In the application:
     ;
     SET KBHERWLM EMPTY;
     ;
     KBHERWLM.SKBH_ELOG_TYPE   = 'MSG';
     KBHERWLM.SKBH_ELOG_PROG   = 'THISPGM';
     KBHERWLM.SKBH_ELOG_SUBRUT = 'MY_PROCESS_IN_ERROR';
     KBHERWLM.SKBH_ELOG_CFROM  = 'TOPPGM';
     KBHERWLM.SKBH_ELOG_OBJECT = XXX.TPNUMBER;
     KBHERWLM.SKBH_ELOG_SUBSYS = 'MY SYS';
     KBHERWLM.SKBH_ELOG_ALERT  = 'N';
     KBHERWLM.SKBH_MSGID       = 'MSG456';
     KBHERWLM.SKBH_MSGVAR1     = XXX.TPNUMBER;
     KBHERWLM.SKBH_MSGVAR2     = 'ACCESS RIGHTS';
     KBHERWLM.SKBH_MSGVAR3     = ' ';
     ;
     ; /* Call common error log
     CALL KBHELGP KBHERWLM (NOMAPS,NONCSP;
     ;
```

If MSG456 has a message text like `'Error updating data for Trading Partner &1 (reason: &2)'` the resulting message in the System Errorlog could be something like: `'Error updating data for Trading Partner 012345678 (reason: ACCESS RIGHTS)'`

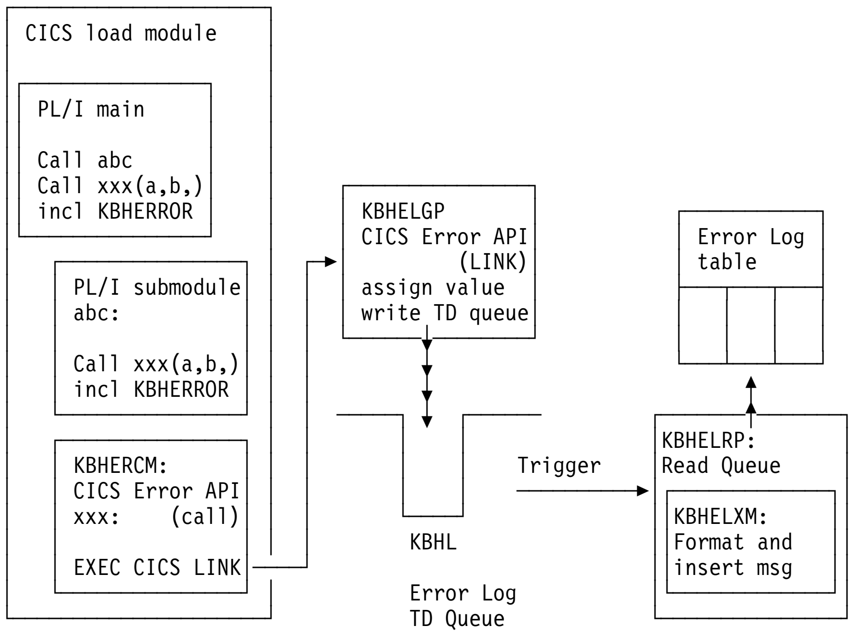
Internal structure

Structure in CICS (CSP)



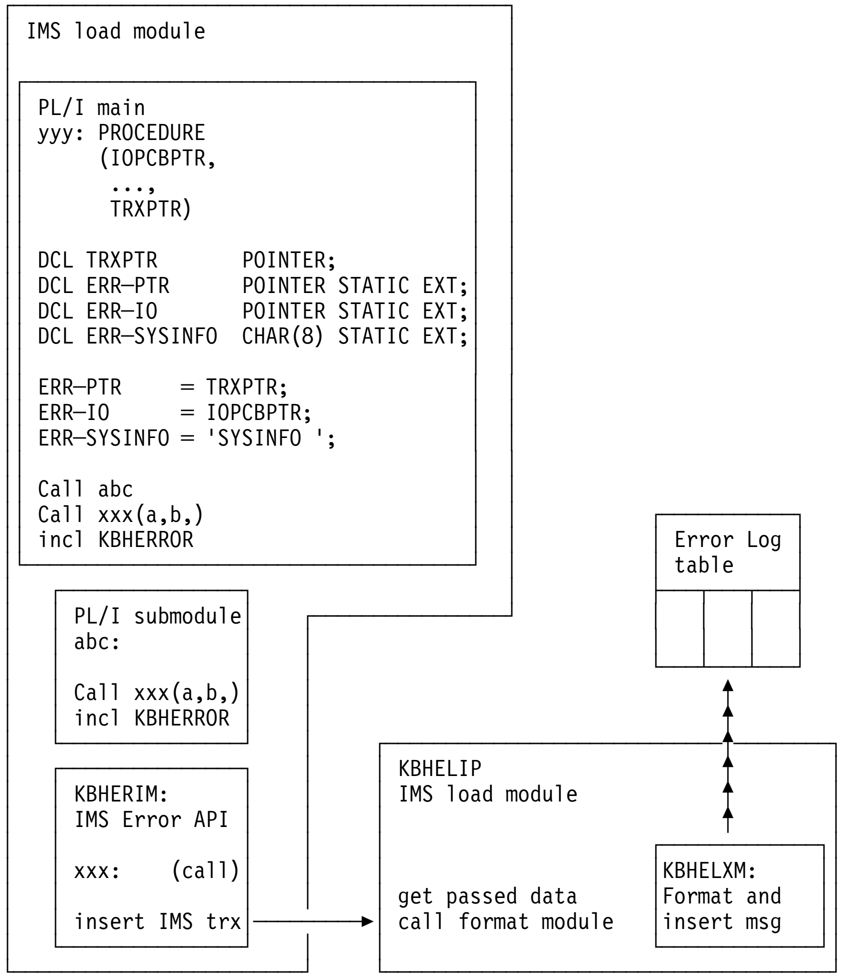
```
Linkage Editor parameters:
No special requirements except linking to a CICS program
```

Structure in CICS (PL/I)



```
Linkage Editor parameters:
INCLUDE SYSNCAL(yourpgm)     Main module in NCAL
INCLUDE SYSNCAL(KBHERCM)     Error log module for CICS
```

Structure in IMS (PL/I)



```
Linkage Editor parameters:
INCLUDE SYSNCAL(yourpgm)     Main module in NCAL
INCLUDE SYSNCAL(KBHERIM)     Error log module for IMS
```

```
PSB definition for IMS transaction:
PCB   ...
PCB   TYPE=TP,NAME=KBHELI0,EXPRESS=YES    Transaction for error program
```

The external variables are declared and initialised in the main program. The external pointer ERR_PTR which is set to the address of the alternate PCB is mandatory, as this is used to make the insert to the IMS queue of the message which triggers the real Error API. The external pointer ERR_IO, which is set to the address of the IO PCB, and the field ERR_SYSINFO belong together, use of one demands the other. These fields are used to extract more information from IMS about the system ID, Lterm, transaction, and so on, where the error occurred. Without these fields the corresponding fields in the ERRLOG are set to blank. In the interests of backward compatibility these variables are not mandatory but it is recommended you use them.

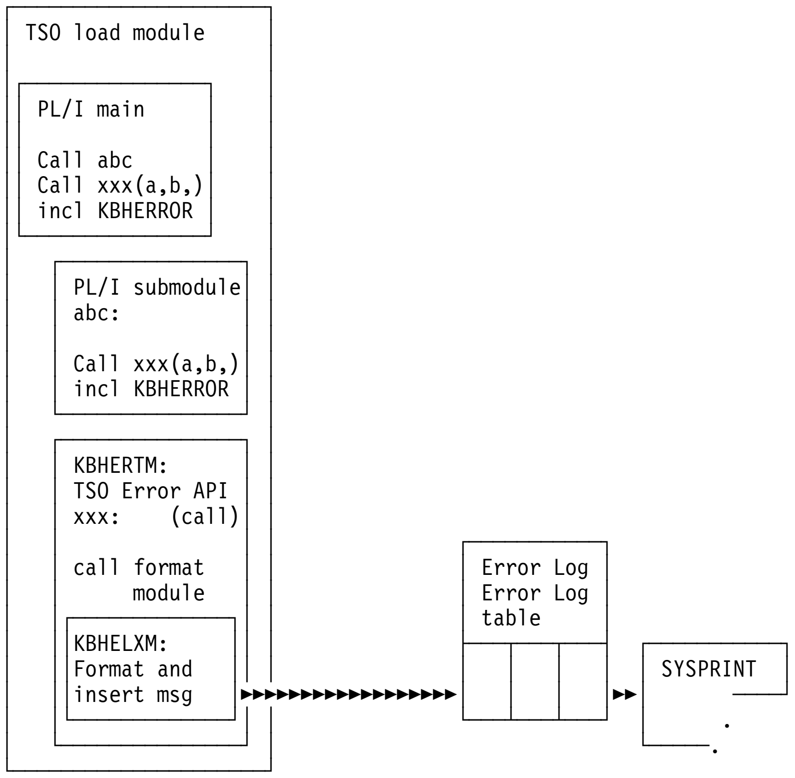
It is important to initialise the external variables at the earliest opportunity in the application program.

The IMS Error transaction is a clone of the CICS Error transaction, and it calls the same internal code to insert the message on the System Errorlog DB2 table. If the

IMS and CICS are running against the same DB2 system then the call will be local. If the IMS and CICS are running on two different DB2 systems, the DB2 release must be release level DB2 3.1 or higher. To perform remote updates, the VTAM application definition for the DB2 systems must be:

```
luname   APPL  APPC=YES,
               ....
               ATTNLOSS=ALL,
               ...
               SYNCLVL=SYNCPT,
               ...
```

Structure in TSO - batch (PL/I)



```
 Linkage Editor parameters:
 INCLUDE SYSNCAL(yourpgm)     Main module in NCAL
 INCLUDE SYSNCAL(KBHERTM)     Error log module for TSO/BATCH with DB2

For applications that only need the errors written to SYSPRINT
with no insertion on the DB2 error log another version exists:

 INCLUDE SYSNCAL(yourpgm)     Main module in NCAL
 INCLUDE SYSNCAL(KBHERBM)     Error log module for TSO/BATCH without DB2
```

Additional features of System Errorlog

Here ar some more features of System Errorlog:

Backward compatibility with old KAAAERR

The old CSP error handler for DB2 errors application KAAAERR has been changed internally to call the new API. As a result, the DBRM KAAAERR must be removed from any application plans.

Backward compatibility with old KBHDB2E
The old PL/I error handler for DB2 errors include member KBHDB2E has been changed internally to call the new API. As a result, the environment specific version of the API must be included using a linkage editor card.

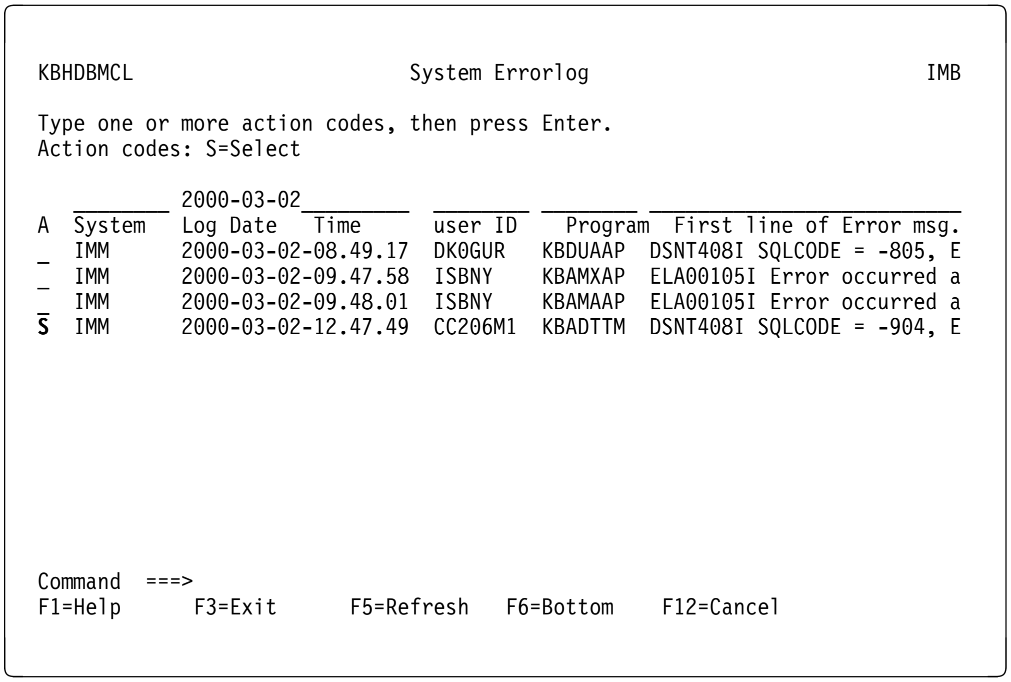
Messages from CSP/RS
CSP/RS puts messages in TD queue ELAD. The System Errorlog has a CICS transaction, that is run by the IMB timer at certain intervals. This transaction will catch these message records and format them into System Errorlog entries. This will assist the error determination, if CSP/RS is involved in an error situation.

Alert scanner
If the call to the System Errorlog API indicated ALERT=Y, then a CICS transaction (run timer driven) will pick up the record later and format the message into a note, and send it to the IMB support person/group (using ISERROR definitions in WC table).

System Errorlog Browse application

An application has been made for the purpose of browsing through the System Errorlog table. The application first displays a selection screen with the possibility of generic search on all of the displayed columns. The selection screen displays all errors matching the search criteria. By selecting a specific error (with a line command) all available information would be displayed in a detail panel.

The application can be called from a command line (fastpath ERRLOG) giving arguments to the search. If no arguments are given, today's date is used as default search criteria.



```
KBHDBMCL                       System Errorlog                          IMB

Type one or more action codes, then press Enter.
Action codes: S=Select

   ________ 2000-03-02_________ ________ ________ __________________________
A  System   Log Date   Time     user ID    Program  First line of Error msg.
_  IMM      2000-03-02-08.49.17  DK0GUR   KBDUAAP  DSNT408I SQLCODE = -805, E
_  IMM      2000-03-02-09.47.58  ISBNY    KBAMXAP  ELA00105I Error occurred a
_  IMM      2000-03-02-09.48.01  ISBNY    KBAMAAP  ELA00105I Error occurred a
S  IMM      2000-03-02-12.47.49  CC206M1  KBADTTM  DSNT408I SQLCODE = -904, E








Command  ===>
F1=Help      F3=Exit      F5=Refresh   F6=Bottom    F12=Cancel
```

Figure 55. Searching in System Errorlog Browse application. You are able to search for a specific Error log registration by typing a search argument, or select more information for an individual row.

Selecting a row on the list will display a panel with more information.

```
KBHDCM1S                 Details of System Errorlog                     IMB


Error Time . . . : 2000-03-02-12.47.49.720000
RACF user ID. . . : CC206M1                     System Id. . . . : IMM
Program. . . . . : KBADTTM                    Transaction. . . : KBA9
Process. . . . . : SQL_FETCH_KBDVTR           Task no. . . . . : 0000105
Object / Table . : KBDVTR                     Terminal . . . . :
Called from. . . :        4                   Sybsystem. . . . : PLI SQL
Message. . . . . : SQL-904                    Severity . . . . : A

Error Message. . :
DSNT408I SQLCODE = -904, ERROR:  UNSUCCESSFUL EXECUTION CAUSED BY AN
         UNAVAILABLE RESOURCE. REASON 00C90082, TYPE OF RESOURCE 00000201, AN
         RESOURCE NAME KBDD001 .KBDXTR
DSNT418I SQLSTATE   = 57011 SQLSTATE RETURN CODE
DSNT415I SQLERRP    = DSNXRRC SQL PROCEDURE DETECTING ERROR
DSNT416I SQLERRD    = 111  13172746  0  13223106  -974970871  12714050 SQL
DIAGNOSTIC INFORMATION

Command  ===>
F1=Help      F3=Exit      F12=Cancel
```

Figure 56. Detailed information in the System Errorlog Browse application.. By pressing F1 online help is available, and the most common DB2 reason codes can be found.

In the case of DB2 errors you can press F1 to get the help panel shown in Figure 57.

Help panel

```
KBHDCM1S              Details of System Errorlog - Help

   From the panel it is possible to see all the detail fields from a
   System Errorlog entry.

   Selection mode
                  All fields are protected.
                  Press F12 to return to list

   Related information
 _ (Common DB2 Resource Types - Help)
 _ (Common DB2 Reason Codes - Help)

   Function keys
     The normal Function keys for a detail panel are valid.

F1=Help      F3=Exit      F12=Cancel
```

Figure 57. Details of System Errorlog - Help panel

Place the cursor next to *Common DB2 Resource Types* and a list of the most common DB2 resource types is displayed, as shown in Figure 58 on page 212.

Common DB2 Resource Types - Help

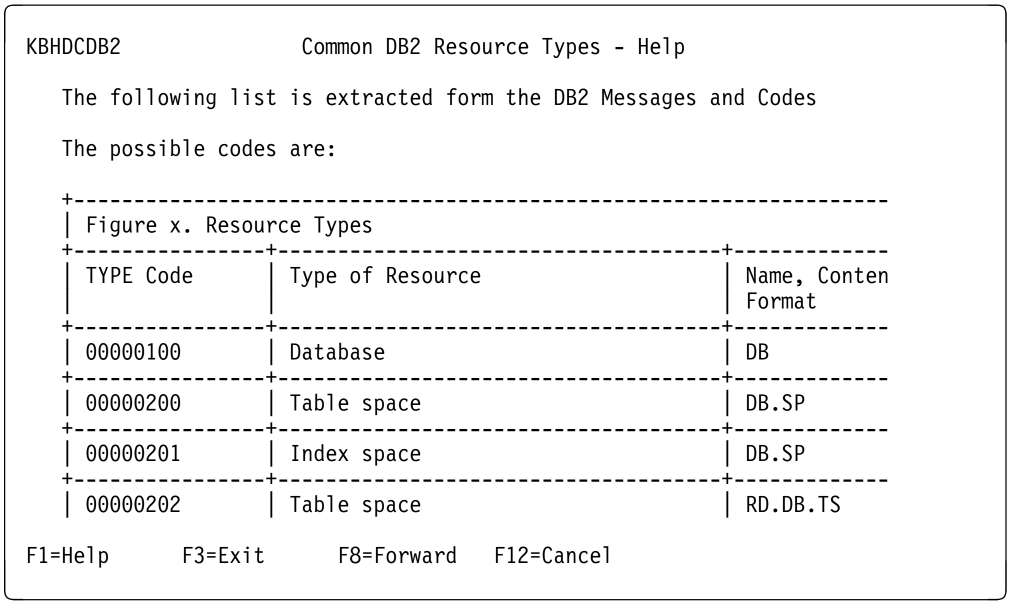

```
KBHDCDB2              Common DB2 Resource Types - Help

   The following list is extracted form the DB2 Messages and Codes

   The possible codes are:

   +---------------------------------------------------------------------
   | Figure x. Resource Types
   +-----------------+--------------------------------------+-------------
   | TYPE Code       | Type of Resource                     | Name, Conten
   |                 |                                      | Format
   +-----------------+--------------------------------------+-------------
   | 00000100        | Database                             | DB
   +-----------------+--------------------------------------+-------------
   | 00000200        | Table space                          | DB.SP
   +-----------------+--------------------------------------+-------------
   | 00000201        | Index space                          | DB.SP
   +-----------------+--------------------------------------+-------------
   | 00000202        | Table space                          | RD.DB.TS

F1=Help      F3=Exit      F8=Forward   F12=Cancel
```

Figure 58. Common DB2 Resource Types - Help panel

Place the cursor next to *Common DB2 Reason codes* and you get a list of the most common DB2 reason codes, as shown in Figure 59.

Common DB2 Reason Codes - Help

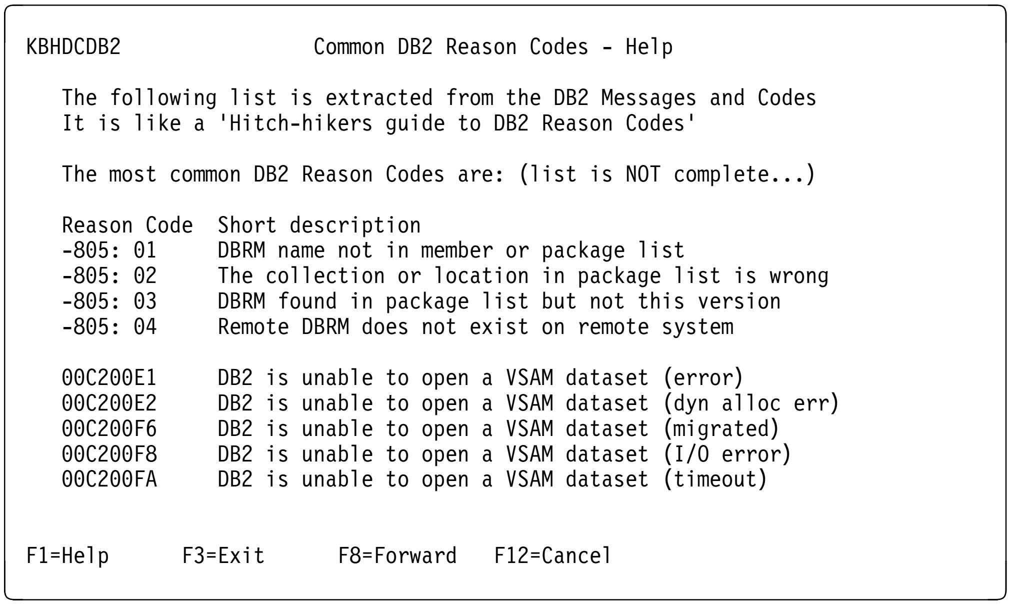

```
KBHDCDB2               Common DB2 Reason Codes - Help

   The following list is extracted from the DB2 Messages and Codes
   It is like a 'Hitch-hikers guide to DB2 Reason Codes'

   The most common DB2 Reason Codes are: (list is NOT complete...)

   Reason Code  Short description
   -805: 01     DBRM name not in member or package list
   -805: 02     The collection or location in package list is wrong
   -805: 03     DBRM found in package list but not this version
   -805: 04     Remote DBRM does not exist on remote system

   00C200E1     DB2 is unable to open a VSAM dataset (error)
   00C200E2     DB2 is unable to open a VSAM dataset (dyn alloc err)
   00C200F6     DB2 is unable to open a VSAM dataset (migrated)
   00C200F8     DB2 is unable to open a VSAM dataset (I/O error)
   00C200FA     DB2 is unable to open a VSAM dataset (timeout)


F1=Help      F3=Exit      F8=Forward   F12=Cancel
```

Figure 59. Common DB2 Reason Codes - Help panel

Determining the cause of the problem

The System Errorlog entry on the previous figure was about a DB2 problem. By using the information available on the panel, it is possible to pinpoint what the exact reason is. The relevant reference manual is the *DB2 Messages and Codes*, where all the necessary information can be found. Extracts of most used codes can also conveniently be found in the online help for the Details of System Errorlog panel, where some common DB2 Resource Types and DB2 Reason Codes are listed.

In this case the problem was a `SQLCODE = -904`, which means that a necessary element in DB2 was not available when (in this case) a process in CICS needed it.

Further the `REASON 00C90082` can explain more: An attempt was made to allocate a resource that is already allocated to a DB2 utility function.

Finally the `TYPE OF RESOURCE 00000201` can tell that the resource is an Index Space with the `RESOURCE NAME KBDD001 .KBDXTR`.

So the basic problem is that a DB2 utility job is using the index KBDXTR exclusively. This index belongs to the MailRoom Transport table KBDTTR which is used by the MailRoom to store documents in. DB2 table maintenance (using DB2 utility jobs) should be performed while the CICS is down to prevent cases like this.

Codepage translation services

Codepage translation is often necessary when sending data from one platform to another. If character data is transmitted from MVS to a PC, codepage translation must be performed from EBCDIC[1] to ASCII[2] before the data is readable on the PC. Both EBCDIC and ASCII exist in a number of variants, or codepages, that are denominated Coded Character Set Identifier (CCSID).

This chapter introduces codepages and discusses common problems when exchanging data in relation to IMB.

EBCDIC: IBM MVS and AS/400 systems use EBCDIC, and codepages are available for several countries. Some samples are:

037	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
277	Denmark and Norway
285	United Kingdom
500	International, Belgium, Canada, Switzerland

EBCDIC codepages represent the letters a—z and A—Z and numbers 0–9 using the same hexadecimal values, but some other characters have different hexadecimal values across codepages.

ASCII: Unix, AIX, PC systems, and the Internet use ASCII. The common ASCII codepages are:

1 EBCDIC is an abbreviation of IBM Extended Binary Coded Decimal Interchange Code.

2 ACSII is an abbreviation of American National Standard Code for Information Interchange.

437	PC Data, USA and many other countries
819	ISO-8859-1
850	PC Data, Europe, Latin countries

The hexadecimal values used for the letters a—z and A—Z and numbers 0–9 are the same in ASCII codepages, but the actual values are not the same as for EBCDIC. Other characters have different hexadecimal values across codepages.

Translation tables

There are many codepages and many combinations of codepage conversions. When performing a conversion from EBCDIC to ASCII it is necessary to have a distinct translation table for each combination and associated conversion logic.

IMB supplies codepage translation tables that are used in different parts of the system, see “Summary of codepage usage” on page 216. The translation tables are listed in Table 14.

Table 14 (Page 1 of 2). SBCS codepage translation tables. IMB supports these single-byte character sets codepage conversions.

Translation table	**From codepage**	**To codepage**
EBCDIC to ASCII translation:		
CP037437	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	PC Data, USA and many other countries
CP037819	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	ISO-8859-1
CP037850	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	PC Data, Europe, Latin countries
CP273819	Germany and Austria	ISO-8859-1
CP277819	Denmark and Norway	ISO-8859-1
CP277850	Denmark and Norway	PC Data, Europe, Latin countries
CP278819	Sweden and Finland	ISO-8859-1
CP280819	Italy	ISO-8859-1
CP284819	Spain	ISO-8859-1
CP285819	United Kingdom	ISO-8859-1
CP297819	France	ISO-8859-1
CP297850	France	PC Data, Europe, Latin countries
CP500819	International, Belgium, Canada, Switzerland	ISO-8859-1
CP500850	International, Belgium, Canada, Switzerland	PC Data, Europe, Latin countries
ASCII to EBCDIC translation:		
CP437037	PC Data, USA and many other countries	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP819037	ISO-8859-1	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP819273	ISO-8859-1	Germany and Austria
CP819277	ISO-8859-1	Denmark and Norway
CP819278	ISO-8859-1	Sweden and Finland
CP819280	ISO-8859-1	Italy
CP819284	ISO-8859-1	Spain

Table 14 (Page 2 of 2). SBCS codepage translation tables. IMB supports these single-byte character sets codepage conversions.

Translation table	From codepage	To codepage
CP819285	ISO-8859-1	United Kingdom
CP819297	ISO-8859-1	France
CP819500	ISO-8859-1	International, Belgium, Canada, Switzerland
CP850037	PC Data, Europe, Latin countries	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP850277	PC Data, Europe, Latin countries	Denmark and Norway
CP850297	PC Data, Europe, Latin countries	France
CP850500	PC Data, Europe, Latin countries	International, Belgium, Canada, Switzerland
EBCDIC to EBCDIC translation:		
CP037280	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	Italy
CP037297	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	France
CP037500	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand	International, Belgium, Canada, Switzerland
CP277500	Denmark and Norway	International, Belgium, Canada, Switzerland
CP280037	Italy	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP280500	Italy	International, Belgium, Canada, Switzerland
CP297037	France	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP297500	France	International, Belgium, Canada, Switzerland
CP500037	International, Belgium, Canada, Switzerland	USA, Canada, Netherlands, Portugal, Brazil, Australia, New Zealand
CP500277	International, Belgium, Canada, Switzerland	Denmark and Norway
CP500280	International, Belgium, Canada, Switzerland	Italy
CP500297	International, Belgium, Canada, Switzerland	France
CP500500	Null conversion	Null conversion

Any EBCDIC to ASCII codepage translation table will convert the letters a—z and A—Z and numbers 0–9 from EBCDIC to ASCII correctly, but many other characters such as national letters and special signs such as @ will be correctly converted only if the exact codepage translation table matching the requirements of the sender and receiver is used.

Codepage translation tables used in IMB have been created using translation tables from the IBM Character Data Representation Architecture (CDRA) Registry. The CDRA roundtrip conversion tables have been used.

Single-byte and double-byte

In European and American countries it is common to represent a character in a single byte, because there are less then 256 different characters. This is called a single-byte character set (SBCS). However, this is not possible in many Asian countries because the large number of characters will not fit into a single byte. It is therefore necessary to use two bytes to represent one character. This is called a double-byte character set (DBCS).

It is relatively simple to perform SBCS conversions, where one character in EBCDIC is translated to the related character in ASCII. In DBCS countries it is common to mix DBCS characters and SBCS characters, called multi-byte character sets (MBCS). In MBCS there is no common way to determine the beginning and end of a DBCS character string. EBCDIC uses *shift-out* and *shift-in* characters. PC systems use the hexadecimal value of a byte to determine if it is SBCS or DBCS. In Internet transmissions it is common to use a three— or four—byte escape character sequence when moving from SBCS to DBCS and back.

The result of a normal SBCS codepage translation is a string that has the same length as the input string. In MBSC this is not the case, because the switch from SBCS to DBCS changes the length of the output string. It is therefore difficult to perform DBCS and MBCS codepage conversion on structured data, because the structure is destroyed due to the length change.

Support of DBCS and MBCS is limited to a few places in IMB, see "Summary of codepage usage." The available MBCS translation tables are listed in Table 15.

Table 15. MBCS codepage translation tables. IMB supports these multi-byte character sets codepage conversions.

Translation table	From codepage	To codepage
Host to Internet translation:		
CP930ISO	Japanese Katakana-Kanji Host Mixed	ISO-2022-JP
CP939ISO	Japanese Latin-Kanji Host Mixed	ISO-2022-JP

Note: The conversion to ISO-2022-JP for Japanese e-mail is based on translation tables from CCSID 930/939 to 5054. A subset of the available SBCS and DBCS charachers are defined in ISO-2022-JP, and it is therefore not recommended to use characters from *JIS X201 Katakana set* and *JIS X212 set* because the receiver e-mail client might not display the characters correctly.

Summary of codepage usage

Codepage conversion is available in a number of places in IMB. All of them support SBCS conversions and a few places have limited support for MBCS conversions.

Mail and Fax API
See "Mail and Fax API—KBHFTXP" on page 222. SBCS codepage conversion from EBCDIC to ASCII and EBCDIC to EBCDIC is possible when sending files and messages.

Internet e-mail API
See "Internet e-mail API—KBHSMTP" on page 232. SBCS codepage conversion from EBCDIC to ASCII is needed for the mail header part. Both SBCS and MBCS is possible for the mail items being sent. The length change is of no importance to e-mails being sent.

MailRoom MQSeries source and destination scenario
See "Using MQSeries MQPUT to send documents to MailRoom" on page 104 and "Using MQSeries MQGET to receive documents from MailRoom" on page 106. Codepage conversions are performed in MQSeries. IMB can calculate the correct length for the transmission length fields, which can be incorrect in the MQSeries codepage conversion due to the length change for MBCS conversions.

MailRoom APPC source scenario

See “APPC MailRoom write/send program—F2A” on page 136. SBCS codepage conversion from ASCII to EBCDIC is performed when receiving the data.

MailRoom APPC destination scenario

See “APPC MailRoom read/receive program—A2F” on page 137. SBCS codepage conversion from EBCDIC to ASCII is performed before sending the data.

MailRoom TCP/IP source scenario

See “TCP/IP MailRoom write/send program—F2T” on page 128 and “OS/2 MailRoom write/send program—FILE2TCP” on page 132. SBCS codepage conversion from ASCII to EBCDIC is performed when receiving the data.

MailRoom TCP/IP destination scenario

See “TCP/IP MailRoom read/receive program—T2F” on page 129 and “OS/2 MailRoom read/receive program—TCP2FILE” on page 133. SBCS codepage conversion from EBCDIC to ASCII is performed before sending the data.

MailRoom Expedite/CICS destination scenario

See also *System Administration Guide*. SBCS codepage conversion from EBCDIC to ASCII is possible before sending the data.

MailRoom codepage conversion exit

See “Codepage conversion exit KBAGXCP” on page 60. SBCS codepage conversion from EBCDIC to EBCDIC is possible in this MailRoom supplied kernel and destination exit.

Validating and calculating dates—KBHDATE

The date API is used to:

ADD	Add a number of days to a date.
CHECK	Check if a date is valid.
DIFF	Calculate the difference between 2 dates
FIRST	Find first month or day.
LAST	Find last month or day.

Format

KBHDATE is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBHDA2P')
                    COMMAREA(COMMAREA)
                    NOHANDLE; ──────────────────────────────────►◄

     DCL 1 COMMAREA,
          2 KBHDATE—PTR     POINTER;
```

Parameters

Parameters as declared in PL/1:

```
/*                                                                */
/*   +---------------------------------------------------------+    */
/*   |              IMB INFRASTRUCTURE                         |    */
/*   |              ===============================            |    */
/*   |                                                         |    */
/*   |  STRUCTURE : KBHDATE                                    |    */
/*   |                                                         |    */
/*   |  LENGTH    : 00500 BYTES                                |    */
/*   |                                                         |    */
/*   |  CONTENTS  : API TO DATE ROUTINE/MODULE                 |    */
/*   |                                                         |    */
/*   |  RELATIONS : NONE.                                      |    */
/*   |                                                         |    */
/*   +---------------------------------------------------------+    */
/*                                                                */
/*                                                         OFFSET */
/*                                                            --- */
    3 SKBH_FNC            CHAR(008), /* FUNCTION NAME           000 */
    3 SKBH_IN_DATE        CHAR(040), /* INPUT DATE 1            008 */
    3 SKBH_IN_DATE2       CHAR(040), /* INPUT DATE 2            048 */
    3 SKBH_IN_FORMAT      CHAR(040), /* FORMAT OF INPUT DATE    088 */
    3 SKBH_OUT_DATE       CHAR(040), /* OUTPUT DATE             128 */
    3 SKBH_OUT_FORMAT     CHAR(040), /* FORMAT OF OUTPUT DATE   168 */
    3 SKBH_PERIOD     FIXED BIN(15), /* DISPLACEMENT OF IN_DATE 208 */
    3 SKBH_PERIOD_TYPE    CHAR(001), /* UNIT OF DISPLACEMENT    210 */
    3 SKBH_MSGID          CHAR(010), /* MESSAGE NUMBER          211 */
    3 RESERV_1            CHAR(279)  /* RESERVED                221 */
/*                                                                */
/*                                           TOTAL LENGTH 00500 */
/*  ==  END OF STRUCTURE KBHDATE  ==                              */
/*  -------------------------------------------------------------- */
```

Here is a description of the fields:

SKBH_FNC
API function, which can contain ADD, CHECK, DIFF, FIRST or LAST.

SKBH_IN_DATE
The input date to be checked or used as base for a calculation.

SKBH_IN_DATE2
Used for second input date in DIFF function.

SKBH_IN_FORMAT
The format of the IN_DATE. The following formats are supported:

TIMESTAMP	DB2 timestamp format: YYYY-MM-DD-HH.MM.SS.mmmmmm
DATE	DB2 ISO format: YYYY-MM-DD
EUR	DD.MM.YYYY format
USA	MM.DD.YYYY format

SKBH_OUT_DATE
The returned date. If the input date is to be checked, no date is returned in this field.

SKBH_OUT_FORMAT
The format of the OUT_DATE. The same formats as for IN_DATE are supported.

SKBH_PERIOD
ADD function: A number (of days) to add to or subtract from the IN_DATE.

DIFF function: Result of the calculation, days between IN_DATE and IN_DATE2

SKBH_PERIOD_TYPE
The measurement of PERIOD. The following kinds are supported:

D: Days

SKBH_MSGID
MailRoom message Id. If the message id is blank, then call was successful.

RESERV_1
Future use. Must be initialised to blanks by calling application.

Examples

Here is an example of calling the API from CSP:

```
;
;/* Call PL1 program KBHDA2P to ADD 8 days to IN_DATE
;
SET KBHDATE EMPTY;
KBHDATE.SKBH_FNC         = 'ADD';
KBHDATE.SKBH_IN_DATE     = '1997-12-24';
KBHDATE.SKBH_IN_FORMAT   = 'DATE';
KBHDATE.SKBH_OUT_FORMAT  = 'DATE';
KBHDATE.SKBH_PERIOD      = 8;
KBHDATE.SKBH_PERIOD_TYPE = 'D';
;
CALL KBHDA2P KBHDATE (NOMAPS,NONCSP;
;
IF KBHDATE.SKBH_MSGID = ' ';
  ;/* OK
  ;/*
  ;/* The date '1998-01-01' is in KBHDATE.SKBH_OUT_DATE
  ;/*
ELSE;
  ;/* Error
  ;/* More info in KBHDATE.SKBH_MSGID
END;
;

;
;/* Call PL1 program KBHDA2P to validate a date in IN_DATE
;
SET KBHDATE EMPTY;
KBHDATE.SKBH_FNC         = 'CHECK';
KBHDATE.SKBH_IN_DATE     = '1997-12-24';
KBHDATE.SKBH_IN_FORMAT   = 'DATE';
;
CALL KBHDA2P KBHDATE (NOMAPS,NONCSP;
;
IF KBHDATE.SKBH_MSGID = ' ';
  ;/* Date OK
  ;/*
ELSE;
  ;/* Error
  ;/* More info in KBHDATE.SKBH_MSGID
END;
;
```

```
;
;/* Call PL1 program KBHDA2P to find first month and day in IN_DATE
;
SET KBHDATE EMPTY;
KBHDATE.SKBH_FNC         = 'FIRST';
KBHDATE.SKBH_IN_DATE     = '1998      ';
KBHDATE.SKBH_IN_FORMAT   = 'DATE';
KBHDATE.SKBH_OUT_FORMAT  = 'DATE';
;
CALL KBHDA2P KBHDATE (NOMAPS,NONCSP;
;
IF KBHDATE.SKBH_MSGID = ' ';
  ;/* Date OK
  ;/*
  ;/* The date '1998-01-01' is in KBHDATE.SKBH_OUT_DATE
  ;/*
ELSE;
  ;/* Error
  ;/* More info in KBHDATE.SKBH_MSGID
END;
;

;
;/* Call PL1 program KBHDA2P to find last day in IN_DATE
;
SET KBHDATE EMPTY;
KBHDATE.SKBH_FNC         = 'LAST';
KBHDATE.SKBH_IN_DATE     = '1998-02   ';
KBHDATE.SKBH_IN_FORMAT   = 'DATE';
KBHDATE.SKBH_OUT_FORMAT  = 'DATE';
;
CALL KBHDA2P KBHDATE (NOMAPS,NONCSP;
;
IF KBHDATE.SKBH_MSGID = ' ';
  ;/* Date OK
  ;/*
  ;/* The date '1998-02-28' is in KBHDATE.SKBH_OUT_DATE
  ;/*
ELSE;
  ;/* Error
  ;/* More info in KBHDATE.SKBH_MSGID
END;
;
```

Processing

The module KBHDA2P will call sub-module KBHDATM which will do the date processing.

KBHDATM can also be used directly from PL/I programs outside the CICS environment, however this method is unsupported, since IMB modules should not be linked into user programs.

Mail and Fax API—KBHFTXP

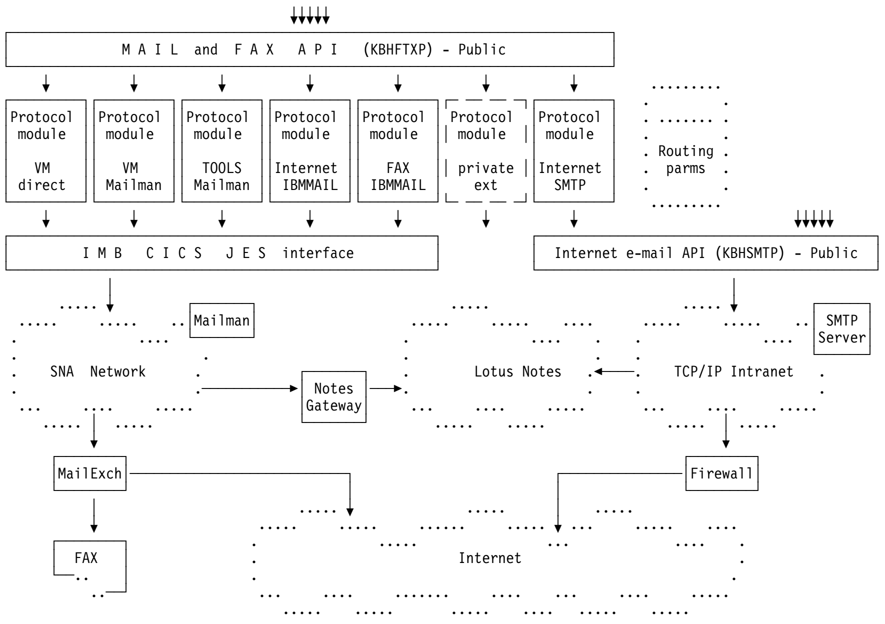
The Mail and Fax API is used to send messages and files to users on different kinds of systems.

Examples are:

- Sending a file to a user on MVS
- Sending a note to a VM user
- Sending e-mail to an Internet user
- Appending to a TOOLS forum
- Sending a fax

The API is very configurable, and various protocol modules are used internally to provide the desired routing to the receiver, see Figure 60. You can also extend the API with private routing modules.

The API can either send the passed data as a file or as a note.



M A I L and F A X A P I (KBHFTXP) - Public
Protocol module VM direct
Protocol module VM Mailman
Protocol module TOOLS Mailman
Protocol module Internet IBMMAIL
Protocol module FAX IBMMAIL
Protocol module private ext
Protocol module Internet SMTP
Routing parms
I M B C I C S J E S interface
Internet e-mail API (KBHSMTP) - Public
Mailman
SNA Network
Notes Gateway
Lotus Notes
TCP/IP Intranet
SMTP Server
MailExch
Firewall
FAX
Internet

Figure 60. Mail and Fax API, protocol modules and routing options.

Format

KBHFTXP is a CICS Main program, with this LINK syntax:

LINK Syntax (PL/1):

```
►►── EXEC CICS LINK PROGRAM('KBHFTXP')
                    COMMAREA(KBHFTW)
                    LENGTH(CSTG(KBHFTW))
                    NOHANDLE; ──────────────────────────────►◄
```

The API also accepts an alternative call method with only a pointer in the communication area. The pointer must then contain the address of the KBHFTW structure. This is convenient when using the API from CSP.

Parameters

Passed structure for the Mail and Fax API. It can be found in KBH.R450.PLINCL(KBHFTW)

```
/* +--------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------     | */
/* | (C) COPYRIGHT IBM DENMARK. 1999.       ALL RIGHTS RESERVED.   | */
/* | (C) COPYRIGHT IBM CORP. 1999.          ALL RIGHTS RESERVED.   | */
/* |                                                               | */
/* |                                                               | */
/* +--------------------------------------------------------------+ */

/*   +----------------------------------------------------------+    */
/*   |              IMB FILE TRANSFER / EMAIL                    |    */
/*   |              ===============================              |    */
/*   |                                                           |    */
/*   |  STRUCTURE : KBHFTW                                       |    */
/*   |                                                           |    */
/*   |  LENGTH    : 00900 BYTES                                  |    */
/*   |                                                           |    */
/*   |  CONTENTS  : CICS API TO SEND MAIL                        |    */
/*   |                                                           |    */
/*   |  RELATIONS : NONE.                                        |    */
/*   |                                                           |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
  3 SKBH_FNC           CHAR(008), /* FUNCTION NAME              000 */
  3 SKBH_MSGID         CHAR(010), /* MESSAGE NUMBER             008 */
  3 IOPUCTY            CHAR(003), /* COUNTRY CODE               018 */
  3 SKBH_LTSQ          CHAR(008), /* LTSQ WITH LARGE DATA       021 */
  3 SKBH_TSQUEUE       CHAR(008), /* SINGLE TS QUEUE WITH DATA  029 */
  3 USERID             CHAR(008), /* USERID OF SENDER           037 */
  3 SKBH_REFERENCE     CHAR(016), /* OPT: REFERENCE             045 */
  3 SKBH_CODEPAGE      CHAR(008), /* OPT: CODEPAGE CONV. TABLE  061 */
  3 KBHFTW_RES1        CHAR(031), /* RESERVED                   069 */
  3 SKBH_MAILBLOCK,               /* MAIL ADRESSING BLOCK       100 */
   4 SKBH_MAILTYPE     CHAR(002), /* TYPE OF ADDRESS            100 */
   4 KBHFTW_RES2       CHAR(008), /* RESERVED                   102 */
   4 SKBH_MAILADR,                /* LONG ADDRESS               110 */
    5 SKBH_MAILADR1    CHAR(016), /* ADDRESS PART 1             110 */
    5 SKBH_MAILADR2    CHAR(016), /* ADDRESS PART 2             126 */
    5 SKBH_MAILADR3    CHAR(016), /* ADDRESS PART 3             142 */
    5 SKBH_MAILADR4    CHAR(016), /* ADDRESS PART 4             158 */
    5 SKBH_MAILADR5    CHAR(016), /* ADDRESS PART 5             174 */
   4 KBHFTW_RES3       CHAR(410), /* RESERVED                   190 */
  3 SKBH_SUBJECT       CHAR(080), /* OPT: USE THIS SUBJECT      600 */
  3 SKBH_SENDER        CHAR(080), /* OPT: SENDER OF THIS MAIL   680 */
  3 SKBH_FILEID,                  /* FILENAME OF FILE TO SEND   760 */
   4 SKBH_FILENAME     CHAR(008), /* EITHER TRADITIONAL 2 PART  760 */
   4 SKBH_FILETYPE     CHAR(008), /* FILENAME.FILETYPE          768 */
   4 KBHFTW_RES4       CHAR(024), /* OR DOTTED NOTATION         776 */
  3 KBHFTW_RES5        CHAR(100)  /* RESERVED                   800 */
/*                                                                   */
/*                                                TOTAL LENGTH 00900 */
/*  ==  END OF STRUCTURE KBHFTW   ==                                 */
/*  ------------------------------------------------------------- */
```

SKBH_FNC

The function code to the API. Possible values:

NOTE	Send the passed data as a note.
FILE	Send the passed data as a file.

SKBH_MSGID

Message id pointing to error message. If the message id is blank, then the call was successful.

IOPUCTY

Country code of calling project. This value are used when determining the routing and looking up default values.

SKBH_LTSQ

Name of the LTSQ containing multiple TS Queues to be sent (support for files). The passed LTSQ is a normal TS Queue where each record is the name of another TS Queue. Multiple TS Queues can be concatenated and sent this way.

Either SKBH_LTSQ or SKBH_TSQUEUE should be used, but not both.

It is the responsibility of the calling program to delete its own TS Queues after the API has been used.

SKBH_TSQUEUE

Name of the TS Queue containing data to be sent.

Either SKBH_TSQUEUE or SKBH_LTSQ should be used, but not both.

It is the responsibility of the calling program to delete its own TS Queues after the API has been used.

USERID

The User ID of the person or system sending a note or a file. In some cases this value is used as the sender identification.

SKBH_REFERENCE

Optional reference key to be used in some mail scenarios (Fax, Mailman, Internet e-mail via SMTP).

SKBH_CODEPAGE

Codepage translation table to be used before sending mail. The usage is dependent on the used mail scenario. Transmission via JES will normally not need conversion, but an EBCDIC to EBCDIC conversion is possible, while Internet e-mail via SMTP needs full EBCDIC to ASCII conversion. Leave blank for scenario dependent default value. See also "Codepage translation services" on page 213.

KBHFTW_RES1

Future use. Must be initiated to blanks by calling program.

SKBH_MAILBLOCK

Structure divided in different fields. Use the following fields for normal mail and simple fax and the KBHFTXW structure for fax via IBM Mail Exchange with full address on cover page.

SKBH_MAILTYPE

The mail type identifies the type of address being used. The meaning of the SKBH_MAILADR1–5 and SKBH_MAILADR is dependent on the mail type. Valid types are:

HO	Host address
VM	VM/Host address
ME	IBMMAIL IEA on IBM Mail Exchange
IN	Internet e-mail address
TL	Tools Append
FX	Telefax number
LI	Distribution list

KBHFTW_RES2
Future use. Must be initiated to blanks by calling program.

SKBH_MAILADR
Normally a structure containing SKBH_MAILADR1–5, but for mail type IN (Internet e-mail) SKBH_MAILADR should be considered one input field containing the e-mail address. See also “Passing different addresses” on page 226.

SKBH_MAILADR1–5
Five fields used to pass different parts of an address to the API. See “Passing different addresses” on page 226 for usage.

KBHFTW_RES3
Future use. Must be initiated to blanks by calling program.

SKBH_SUBJECT
Free text to appear in the subject of the mail. Primarily intended for notes, but also used for files sent to Internet via SMTP.

SKBH_SENDER
Optional address of sender. If not specified here, the API will lookup a default sender based on the country code. The format of the field is: name name name   User ID at Node ID
It is important that either at, At or AT is used between the User ID the Node ID.

SKBH_FILEID
Pass a full filename when sending a file. Use SKBH_FILEID as one long field if the file should be named PROJECT.TEST.DATA

SKBH_FILENAME
Or pass the filename TEST in SKBH_FILENAME

SKBH_FILETYPE
And pass the filetype DATA in SKBH_FILETYPE

KBHFTW_RES4
And leave this field blank.

KBHFTW_RES5
Future use. Must be initiated to blanks by calling program.

Processing

The Mail and Fax API is called with an address and some data to send. Depending on the used country code **IOPUCTY** the API will select the proper routing module and transmit the passed data to the receiver.

The routing can be configurable using entries in the working criteria (WC) table, see *IMB 4.5: Installation Guide*. So, you can send files or notes direct to the receiver or send it via a mailman. Furthermore e-mail for Internet can be sent via the Internet e-mail API and a SMTP server, or via a more simple path via IBM Mail Exchange.

Passing different addresses

The API must be called with a mail type and an associated address. The API supports the mail types and addressing format shown in Table 16.

Table 16. Mail addressing modes. The supported mail types and associated usage of **SKBH_MAILADR1–5**

Mail type	**Description and usage**				
	SKBH_MAILADR1	**SKBH_MAILADR2**	**SKBH_MAILADR3**	**SKBH_MAILADR4**	**SKBH_MAILADR5**
HO	Host address: *user ID at node ID*				
	`User ID`	`Node ID`	(not used)	(not used)	(not used)
VM	VM/Host address: *user ID at node ID* VM is usually an alias for HO, but can be routed differently.				
	`User ID`	`Node ID`	(not used)	(not used)	(not used)
ME	IBMMAIL IEA on IBM Mail Exchange: *user ID at IBMMAIL*				
	`User ID`	`IBMMAIL`	(not used)	(not used)	(not used)
IN	Internet e-mail address: *j_doe@somedomain.com*. The e-mail address can span SKBH_MAILADR1–5				
TL	Tools append to FORUM: *my_forum FORUM on disk*				
	`forum name`	`forum type`	`forum disk`	(not used)	(not used)
FX	Telefax: *cty_code + fax_number* Use record overlay KBHFTWF to specify information for cover page. See “IBM Mail Exchange Fax record overlay.”				
	`fax number`	`country code`	(not used)	(not used)	(not used)
LI	Distribution list *DIST-xxxxxxxx*				
	`list name xxxxxxxx`	(not used)	(not used)	(not used)	(not used)

IBM Mail Exchange Fax record overlay

When sending a fax through IBM Mail Exchange you can specify additional information to appear on the fax cover page. This is done by using a record overlay KBHFTWF and filling in the **SKBH_FAXADR** structure with information about the receiver and sender of the fax.

Alternative structure for the Mail and Fax API for sending a fax to IBM Mail Exchange. It can be found in KBH.R450.PLINCL(KBHFTWF)

```
⋮
   3 SKBH_MAILBLOCK,                /* MAIL ADRESSING BLOCK       100 */
    4 SKBH_MAILTYPE      CHAR(002), /* TYPE OF ADDRESS            100 */
    4 KBHFTW_RES2        CHAR(008), /* RESERVED                   102 */
    4 SKBH_FAXADR,                  /* FAX ADDRESSING SUB STRUCT  110 */
     5 SKBH_FAX_NO        CHAR(016),/* FAX TELEPHONE NUMBER       110 */
     5 SKBH_FAX_CTY       CHAR(016),/* TELEPHONE COUNTRY CODE     126 */
     5 SKBH_FAX_TO_NAME   CHAR(060),/* ATTENTION PERSON           142 */
     5 SKBH_FAX_TO_TP     CHAR(060),/* TO COMPANY                 202 */
     5 SKBH_FAX_TO_TPADR1 CHAR(060),/* TO ADDRESS LINE1           262 */
     5 SKBH_FAX_TO_TPADR2 CHAR(060),/* TO ADDRESS LINE2           322 */
     5 SKBH_FAX_TO_TPZIP  CHAR(015),/* TO ZIP NUMBER              382 */
     5 SKBH_FAX_TO_TPCITY CHAR(045),/* TO CITY                    397 */
     5 SKBH_FAX_FROM_NAME CHAR(060),/* FROM PERSON/SYSTEM         442 */
     5 SKBH_FAX_FROM_TP   CHAR(060),/* FROM COMPANY               502 */
    4 KBHFTWF_RES3        CHAR(038),/*                            562 */
   3 SKBH_SUBJECT
⋮
```

SKBH_MAILBLOCK
Structure divided in different fields. Use the following fields only when sending a fax to IBM Mail Exchange

SKBH_MAILTYPE
The mail type identifies the type of address being used. For sending a fax to IBM Mail Exchange, it is necessary to use type **FX**

KBHFTW_RES2
Future use. Must be initiated to blanks by calling program.

SKBH_FAXADR
Address structure for formatting a fax cover page for IBM Mail Exchange

SKBH_FAX_NO
The fax telephone number to send the fax to.

SKBH_FAX_CTY
The international telephone country code

SKBH_FAX_TO_NAME
Name of the person to receive the fax

SKBH_FAX_TO_TP
Company name of receiver

SKBH_FAX_TO_TPADR1
Address line 1 of receiver

SKBH_FAX_TO_TPADR2
Address line 2 of receiver

SKBH_FAX_TO_TPZIP
ZIP number of receiver

SKBH_FAX_TO_TPCITY
City name of receiver

SKBH_FAX_FROM_NAME
Name of person sending the fax

SKBH_FAX_FROM_TP
company name of sender

KBHFTW_RES3

Future use. Must be initiated to blanks by calling program.

Examples

Small example of calling the API from PL/1:

```
KBHFTW = '';

KBHFTW.SKBH_FNC      = 'NOTE';
KBHFTW.IOPUCTY       = '123';      /* My country code         */
KBHFTW.USERID        = 'CICSUSER'; /* Userid of the sender    */
KBHFTW.SKBH_TSQUEUE  = 'ENGLISH';  /* TS Queue with data      */
KBHFTW.SKBH_MAILTYPE = 'HO';       /* Address is MVS host     */
KBHFTW.SKBH_MAILADR1 = 'MYUSER';   /* userid on host system   */
KBHFTW.SKBH_MAILADR2 = 'MYNODE';   /* nodeid of host system   */
KBHFTW.SKBH_SUBJECT  = 'This is the subject of my note';
                       /* Finally override the default sender */
KBHFTW.SKBH_SENDER   = 'Santa Claus    SANTA at GREENL';


/* Call the API */
EXEC CICS LINK PROGRAM('KBHFTXP')
               COMMAREA(KBHFTW)
               LENGTH(CSTG(KBHFTW))
               NOHANDLE;

/* Check return code */
IF EIBRESP = 0 THEN
  IF KBHFTW.SKBH_MSGID = '' THEN
    PUT SKIP EDIT('e-mail was sent OK') (A);
  ELSE
    PUT SKIP EDIT('e-mail was not sent. Message=',
                  KBHFTW.SKBH_MSGID) (A,A);
ELSE
  PUT SKIP EDIT('Problem during link to API. EIBRESP=',
                EIBRESP,' EIBRESP2=',EIBRESP2) (A,F(4),A,F(4));
```

Send File to user panel—KBHSFAP

When an application presents information to a user, it might also need to provide a facility to send the same information to the user for further processing. This module provides a method to let the user specify where the data should be sent, as e-mail or as a file. If the user at a later time in current session enters the application, the application remember the address type and address.

The module is for CSP applications running under IMB control.

Format

A CICS version of this module is available.

Call Syntax (CSP)

```
►►──CALL KBHSFAP KBHSFWA,KAAWCOM;──────────────────────────────►◄
```

```
CSP Working storage KBHSFWA:

NAME              LEVEL OCCURS TYPE LNG  DESCRIPTION
FUNCTION           10       1  CHA    6  function to be performed
  WHO              20       1  CHA    3  Who are we: CEN(tral)/LOC(al)
  WHAT             20       1  CHA    3  What to do: SND
SKBH_FNC           10       1  CHA    8  Function name
SKBH_FILEID        10       1  CHA   40  File name long version
SKBH_FILENAME      10       1  CHA    8  File name (part 1)
SKBH_FILETYPE      10       1  CHA    8  File type (part 2)
SKBH_MAILTYPE      10       1  CHA    2
SKBH_SUBJECT       10       1  CHA   80  Optional: Subject of mail
SKBH_DELETE_TSQ    10       1  CHA    1  Should TS Queues be deleted
SKBH_LTSQ          10       1  CHA    8  List TS Queue (multiple TSQs)
SKBH_TSQUEUE       10       1  CHA    8  TS Queue name
SKBH_REFERENCE     10       1  CHA   16  Optional reference
SKBH_MAILADR       10       1  CHA   80  Mail address
*                  10       1  CHA  135  Reserved
```

WHO A code to be passed on input to indicate who the caller is. Valid values are:

CEN to indicate that the user is free to type any electronic address

LOC to indicate that the user must select an address from a list of electronic addresses defined for the Trading Partner.

WHAT
A code to indicate what is to be done by the application. The value should be *SND* to indicate send function.

SKBH_FNC
An optional function code to be passed. It can be used to override the Send mode field on the panel. Valid values are blank, *FILE* and *NOTE*. The default is to send the passed data as a file.

SKBH_FILEID
Together with SKBH_FILENAME and SKBH_FILETYPE this field can be used to pass a predefined value to appear in the Filename field on the panel. The user can update this panel field.

The SKBH_FILEID is a full file name in the format *MYFILE.TXT*.

Leave SKBH_FILENAME and SKBH_FILETYPE blank if using SKBH_FILEID.

SKBH_FILENAME

Together with SKBH_FILETYPE and SKBH_FILEID this field can be used to pass a predefined value to appear in the Filename field on the panel. The user can update this panel field.

The SKBH_FILENAME is part 1 of a full filename: FILENAME.FILETYPE

Use SKBH_FILETYPE together with SKBH_FILENAME and leave SKBH_FILEID blank.

SKBH_FILETYPE

Together with SKBH_FILENAME and SKBH_FILEID this field can be used to pass a predefined value to appear in the Filename field on the panel. The user can update this panel field.

The SKBH_FILETYPE is part 2 of a full filename: FILENAME.FILETYPE

Use SKBH_FILENAME together with SKBH_FILETYPE and leave SKBH_FILEID blank.

SKBH_MAILTYPE

Use this field to pass a preselected electronic address type to be used on the panel. For example, *IN* for an Internet e-mail address. This will cause the panel to start with a single long input field instead for the default user ID and node ID fields for usual host addresses. Leave the field blank for default type.

SKBH_SUBJECT

Use this field to pass a preselected subject to appear on the panel.

SKBH_DELETE_TSQ

Pass value *Y* to ask for deletion of passed TS queues after the data has been sent.

SKBH_LTSQ

Pass the name of a List TS Queue (LTSQ) containing data in one or multiple TS queues to be sent.

Use either SKBH_LTSQ or SKBH_TSQUEUE to pass data, but not both.

SKBH_TSQUEUE

Pass the name of a single TS Queue containing data to be sent.

Use either SKBH_LTSQ or SKBH_TSQUEUE to pass data, but not both.

SKBH_REFERENCE

Optionaly pass a value or reference identifying the data to be sent. It is used for some types of electronic addresses.

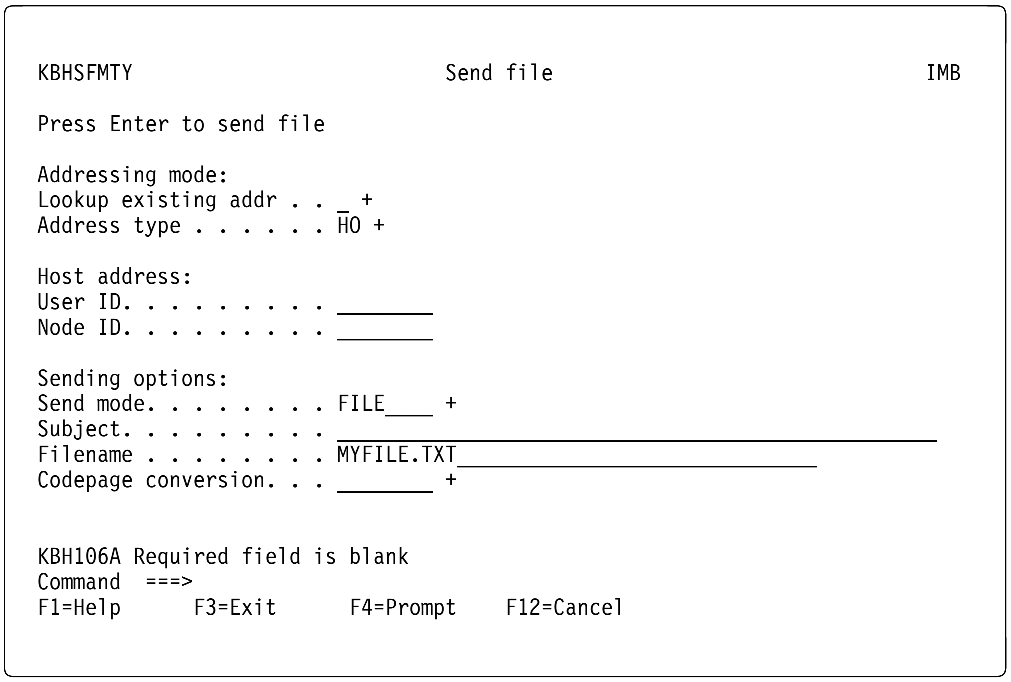
SKBH_MAILADR

A value returned after successful processing to tell where the data was actually sent.

After a call to KBHSFAP field ZMSGNO in structure KAAWCOM will contain message number KBH352 to indicate that the data was sent successfully, or other message numbers to indicate errors.

Processing

The calling application must prepare the data to be sent in either a single TS Queue, or in List TS Queue (LTSQ) (see: “Multiple TS queues” on page 21) if the data is too big for one TS Queue. Then the KBHSFAP CSP application should be called. It will present the user with the panel shown in Figure 61. KBHSFAP validates the values entered by the user and performs the send function.



```
KBHSFMTY                          Send file                             IMB

Press Enter to send file

Addressing mode:
Lookup existing addr . . _ +
Address type . . . . . . HO +

Host address:
User ID. . . . . . . . . ________
Node ID. . . . . . . . . ________

Sending options:
Send mode. . . . . . . . FILE____ +
Subject. . . . . . . . . __________________________________________________
Filename . . . . . . . . MYFILE.TXT______________________________
Codepage conversion. . . ________ +


KBH106A Required field is blank
Command  ===>
F1=Help      F3=Exit      F4=Prompt    F12=Cancel
```

Figure 61. Send file panel

Examples

How to use KBHSFAP in a CSP application:

```
;
SET KBHSFWA EMPTY;      /* Always initialize WS
;
KBHSFWA.WHO = 'CEN';    /* User is allowed to type any address
KBHSFWA.WHAT = 'SND';   /* We need send function
KBHSFWA.SKBH_FILETYPE = 'MYFILE';
KBHSFWA.SKBH_MAILTYPE = 'TXT';
KBHSFWA.SKBH_SUBJECT = "Data export XYZ from Whatever system";
KBHSFWA.SKBH_DELETE_TSQ = 'Y';
KBHSFWA.SKBH_TSQUEUE = MY_DATA_TSQ;
;
CALL KBHSFAP KBHSFWA,KAAWCOM; /* Call module
;
IF ZMSGNO = 'KBH352';   /* OK file was sent
  ; /* You might keep KBHSFWA.SKBH_MAILADR to tell where data was sent
ELSE;
  ; /* We have a problem
END;
;
```

Internet e-mail API—KBHSMTP

You can use the Internet e-mail API to send messages and files to users Internet users via the Simple Mail Transfer Protocol (SMTP) for e-mail. The specifications of SMTP can be found in RFC[3] 821.

Messages and files are passed to the API as CICS TS Queues and then transmitted to Internet through TCP/IP communication with an SMTP server.

The API implements Multipurpose Internet Mail Extensions (MIME), which enables it to send complex e-mails with both a plain message as well as attached files. The specifications of MIME can be found in RFC 2045 - 2049.

The API is called in different ways to either send a mail with a single message or an attached file, to send a mail with up to five messages or attachments, or to send an unlimited number[4] of messages or attachments.

Format

KBHSMTP is a CICS Main program, with this LINK syntax:

LINK Syntax (PL/1):

```
▸▸── EXEC CICS LINK PROGRAM('KBHSMTP')
                    COMMAREA(KBHSMWT)
                    LENGTH(CSTG(KBHSMWT))
                    NOHANDLE; ──────────────────────────▸◂
```

The API also accepts an alternative call method with a only a pointer in the communication area. The pointer must then contain the address of the KBHSMWT structure. This is convenient when using the API from CSP.

Parameters

Passed structure for the Internet e-mail API. It can be found in KBH.R450.PLINCL(KBHSMWT)

```
/* +-----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------       | */
/* | (C) COPYRIGHT IBM DENMARK. 1999.      ALL RIGHTS RESERVED.      | */
/* | (C) COPYRIGHT IBM CORP.    1999.      ALL RIGHTS RESERVED.      | */
/* |                                                                 | */
/* |                                                                 | */
/* +-----------------------------------------------------------------+ */

/*   +-----------------------------------------------------------+    */
/*   |              IMB TRANSMIT FILE/NOTE VIA SMTP               |    */
/*   |              ==================================            |    */
```

3 The name of the result and the process for creating a standard on the Internet. New standards are proposed and published online, as a *Request For Comments*. The Internet Engineering Task Force is a consensus-building body that facilitates discussion, and eventually a new standard is established, but the reference number and name for the standard retains the acronym *RFC*, for example, the official standard for e-mail is RFC 822.

4 While the API has no coded limits on the number of attachments, it can be be limited by the total size of the e-mail being sent. Mail gateways have implemented various size limits (for example 2 megabytes or 4 megabytes) to avoid abuse of the e-mail system (like sending very big files as e-mails).

```
/*   |                                                          |    */
/*   |  STRUCTURE : KBHSMWT                                     |    */
/*   |                                                          |    */
/*   |  LENGTH    : 2000 BYTES                                  |    */
/*   |                                                          |    */
/*   |  CONTENTS  : INTERFACE TO KBHSMTM                        |    */
/*   |                                                          |    */
/*   |  RELATIONS : NONE.                                       |    */
/*   |                                                          |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
    3 SKBH_FNC             CHAR(8),    /* FUNCTION TO PERFORM   000 */
    3 SKBH_MSGID           CHAR(10),   /* MESSAGE NUMBER        008 */
    3 SKBH_BASE_CODEPAGE   CHAR(8),    /* CODEPAGE FOR MAIL HDR 018 */
    3 SKBH_SUBJECT         CHAR(80),   /* SUBJECT OF MAIL       026 */
    3 SKBH_SENDER          CHAR(80),   /* SENDER OF THIS MAIL   106 */
    3 SKBH_REPLY_TO        CHAR(80),   /* OPTIONAL REPLY TO     186 */
    3 SKBH_REFERENCE       CHAR(16),   /* OPTIONAL REFERENCE    266 */
    3 SKBH_MULTIX_TSQUEUE  CHAR(8),    /* MULTIX SESSION TSQ    282 */
    3 SKBH_SMTP_SERVER     CHAR(80),   /* SMTP SERVER HOSTNAME  290 */
    3 SKBH_SMTP_PORT       CHAR(5),    /* SMTP SERVER PORTNUM.  370 */
    3 IOPUCTY              CHAR(3),    /* COUNTRY CODE          375 */
    3 KBHSMWT_RES1         CHAR(22),   /* RESERVED              378 */
    3 SKBH_INET_RECV(5),               /* LIST OF RECEIVERS     400 */
      5 SKBH_RECV_TYP      CHAR(2),    /* RECEIVER TYPE TO/CC  *400 */
      5 SKBH_MAILADR       CHAR(80),   /* RECEIVER E-MAIL ADR  *402 */
      5 KBHSMWT_RES2       CHAR(18),   /* RESERVED             *482 */
    3 SKBH_MAIL_ITEM(5),               /* CODEPAGE CONV         900 */
      5 SKBH_TSQ_MODE      CHAR(2),    /* TYPE OF TS QUEUE     *900 */
      5 SKBH_TSQUEUE       CHAR(8),    /* TS QUEUE WITH DATA   *902 */
      5 SKBH_ITEM_TYPE     CHAR(8),    /* TYPE OF ITEM         *910 */
      5 SKBH_CODEPAGE      CHAR(8),    /* CODEPAGE CONV        *918 */
      5 SKBH_FILEID        CHAR(40),   /* EXTERNAL FILENAME    *926 */
      5 SKBH_MIME_TYPE     CHAR(40),   /* MIME TYPE            *966 */
      5 SKBH_MIME_ENCOD    CHAR(8),    /* MIME ENCODING       *1006 */
      5 KBHSMWT_RES3       CHAR(36),   /* RESERVED            *1014 */
    3 KBHSMWT_RES4         CHAR(350)   /* RESERVED             1750 */
/*                                                TOTAL LENGTH  2000 */
/*                                                                   */
/*  ==  IMB  == END OF STRUCTURE KBHSMWT ==                          */
/*  ---------------------------------------------------------------- */
/*                                                                   */
```

SKBH_FNC

The function code to the API. Possible values:

SINGLE	Send mail item one to the listed receivers.
MULTI	Send up to five mail items to the listed receivers.
MULTIX	Prepare five mail items for later sending.
SENDX	Send previously prepared (one or more MULTIX calls) mail items to the listed receivers.

SKBH_MSGID

Message id pointing to error message. If the message id is blank, then the call was successful.

SKBH_BASE_CODEPAGE
: Reference to a codepage translation table. This translation table will be used for all the mail headers and for other mail items unless overwritten on the individual mail item. This translation table must be single byte only. See “Codepage translation services” on page 213.

SKBH_SUBJECT
: Free text to appear in the subject of the mail.

SKBH_SENDER
: Optional e-mail address of sender. If not specified here, the API will lookup a default sender based on the country code. See “Valid e-mail addresses” on page 238 for valid formats.

SKBH_REPLY_TO
: Optional e-mail address. If specified here, the mail will have a `Reply-To:` mail header. See “Valid e-mail addresses” on page 238 for valid formats.

SKBH_REFERENCE
: Optional text to appear in an additional mail header `X-IMB-Ref:`.

SKBH_MULTIX_TSQUEUE
: Intermediate TS Queue used internally in a sequence of MULTIX, MULTIX, ..., SENDX calls to hold the mail items to be sent. The field must be blank on first call and must not be changed on subsequent calls.

  This TS Queue will be deleted by the API after a call with function code SENDX or after an unsuccessfull MULTIX call.

SKBH_SMTP_SERVER
: Optional hostname or IP address of the SMTP server. If not specified here, the API will lookup a default server based on the country code.

SKBH_SMTP_PORT
: Optional port number of the SMTP server. The commonly used port number for SMTP servers are 00025. If not specified here, the API will lookup a default port number based on the country code.

IOPUCTY
: Country code of calling project. This value are used when looking up default values.

KBHSMWT_RES1
: Future use. Must be initiated to blanks by calling program.

SKBH_INET_RECV
: A structure containing up to five receivers of the e-mail. More receivers can be specified by using a TS Queue.

SKBH_RECV_TYP
: The type of receiver:

  TO Normal To receiver

  CC Receiver is Cc (carbon-copy)

  BC Receiver is Bcc (blind carbon-copy)

  TS Indicate that SKBH_MAILADR is the name of a TS Queue containing more receivers. The format of records in such a TS Queue is 2 byte receiver type and 80 byte e-mail address.

SKBH_MAILADR
The e-mail address of a receiver. See "Valid e-mail addresses" on page 238 for valid formats of an e-mail address.

A total of 100 receivers are supported by the API.

KBHSMWT_RES2
Future use. Must be initiated to blanks by calling program.

SKBH_MAIL_ITEM
A structure containing up to five mail items to be sent. Item one is sent with function SINGLE, all five are sent with function MULTI. More mail items can be queued for sending by using function MULTIX one or more times folowed by a call with function SENDX.

SKBH_TSQ_MODE
The mode of passing data in TS Queues:

TS	The data is passed in a normal TS Queue (default).
LT	The data is passed in an LTSQ (support for big items). The passed LTSQ is a normal TS Queue where each record is the name of another TS Queue. Multiple TS Queues can be concatenated and sent this way.

SKBH_TSQUEUE
Name of the TS Queue (or LTSQ) containing data to be sent.

It is the responsibility of the calling program to delete its own TS Queues after the e-mail has been sent.

SKBH_ITEM_TYPE
Identification of the item to be sent

MAIL	Use data as a mail message (default). Perform codepage conversion.
FILE	Use data as a file attachment. Perform codepage conversion.
FILEBIN	Use data as a binary file attachment. No codepage conversion.
RAW	Pass data as-is to SMTP. No codepage conversion.

SKBH_CODEPAGE
Codepage translation table to be used for current mail item. If not specified the base codepage will be used. See "Codepage translation services" on page 213.

SKBH_FILEID
A end-user filename must be passed here for file attachments

SKBH_MIME_TYPE
The MIME content type for mail item. Leave blank for default values for different mail item types.

text/plain
application/octet-stream
image/gif
image/jpeg
Other values as specified in RFC 2046

SKBH_MIME_ENCOD
The MIME content transfer encoding be performed. Leave blank for default values for different mail item types. Valid values:

QUOTED	Send as quoted printable. Special characters are encoded to prevent transmission problems.
7BIT	Send as 7 bit. Only usable for US ASCII. Line length below 998 bytes.
8BIT	Send as 8 bit. Special characters are sent as-is. Line length below 998 bytes.
BINARY	Send as binary (as-is), no special encoding.
BASE64	Send as base 64 encoded data. Robust transmission.

KBHSMWT_RES3
Future use. Must be initiated to blanks by calling program.

KBHSMWT_RES4
Future use. Must be initiated to blanks by calling program.

Processing

The Internet e-mail API can either be called with just a few fields filled in and taking advantage of the default values, or it can be called with all fields used to obtain special results.

Normally an e-mail is just a plain message, or it is one message with one or more attached files. But other combinations are also possible. The mail item type **SKBH_ITEM_TYPE** is used to control how the items passed to the API are used in the sent e-mail.

Item type—MAIL

An item of type MAIL is interpreted as a message, that is directly readable in the sent e-mail. The first item should usually be such a message.

Data is passed in a TS queue. Every record will become a line in the mail. Record width up to 2000 bytes is supported by the API but not necessarily by all mail gateways. The subject is not part of the message, it must be passed directly in **SKBH_SUBJECT** since it is used in the mail headers.

The default MIME content type **SKBH_MIME_TYPE** for this item is text/plain and the MIME content transfer encoding type **SKBH_MIME_ENCOD** is QUOTED.

Codepage translation is always performed based on the contents of **SKBH_CODEPAGE**. Both single byte codepage conversion and double byte conversion is possible via various codepage translation tables. See “Codepage translation services” on page 213 for more information. If the message is double byte character set it is necessary to override the MIME content transfer encoding type **SKBH_MIME_ENCOD** to 7BIT.

The end-user filename **SKBH_FILEID** is not used for this item type.

Item type—FILE

An item of type FILE will appear in the e-mail as an attached file. The MIME content disposition: attachment header is used to achieve this. While it is possible to send an e-mail with just an attached file and no message, it is not common practice.

Data is passed in a TS queue. Every record will become a line in the attached file (CR+LF is added at the end of a record). Record width up to 2000 bytes is supported by the API but not necessarily by all mail gateways.

The default MIME content type **SKBH_MIME_TYPE** for this item is text/plain and the MIME content transfer encoding type **SKBH_MIME_ENCOD** is 8BIT. It is possible to use BASE64 encoding to avoid possible reformatting by a mail gateway on the way to the receiver.

Codepage translation is always performed based on the contents of **SKBH_CODEPAGE**. Both single byte codepage conversion and double byte conversion is possible via various codepage translation tables. See “Codepage translation services” on page 213 for more information.

The end-user filename **SKBH_FILEID** can be used to give a meaningful filename like MYFILE.TXT for this item type. The receiver will use this filename when detaching the file.

Item type—FILEBIN

An item of type FILEBIN will appear in the e-mail as an attached file. The MIME content disposition: attachment header is used to achieve this. While it is possible to send an e-mail with just an attached file and no message, it is not common practice.

Binary data is passed in a TS queue. While binary data is interpreted as a long stream of bytes, it must be blocked in records in order to be stored in a TS Queue. Record width up to 2000 bytes is supported by the API.

The default MIME content type **SKBH_MIME_TYPE** for this item is application/octet-stream and the MIME content transfer encoding type **SKBH_MIME_ENCOD** is BASE64. It is not recommended to use a different encoding type, but the content type could be changed to image/gif or image/jpeg for graphic images.

Codepage translation is never performed for this item type.

The end-user filename **SKBH_FILEID** can be used to give a meaningful filename like MYLOGO.JPG for this item type. The receiver will use this filename when detaching the file.

Item type—RAW

An item of type RAW can be used if a program needs to specify other MIME headers not directly supported by the API. The caller must supply all necessary MIME headers except the beginning and ending boundary, which is written by the API.

ASCII data including CR+LF is passed in a TS queue. Record width up to 2000 bytes are supported by the API.

The fields for MIME content type, MIME encoding type, codepage translation and filename are not used for this item type.

Valid e-mail addresses

E-mail addresses are passed to the API in three fields: sender, receiver, and optional reply-to field.

Valid formats are:

j_doe@somedomain.com
<j_doe@somedomain.com>
"John Doe" <j_doe@somedomain.com>

The *at sign* @ is known to cause problems in the EBCDIC world because it has different hexadecimal values in different EBCDIC codepages. The API will recognize a serie of values as a valid @ and convert correctly before sending. This way an English user can type an address on the terminal and it will still be valid for a French user.

Examples

Small example of calling the API from PL/1:

```
KBHSMWT = '';
KBHSMWT.SKBH_FNC = 'MULTI';
KBHSMWT.IOPUCTY  = '000';
/* Conversion from Internatl. EBCDIC (CP500) to ISO-8859-1 (CP819) */
KBHSMWT.SKBH_BASE_CODEPAGE = 'CP500819';
KBHSMWT.SKBH_SUBJECT = 'This is my subject';
KBHSMWT.SKBH_RECV_TYP(1)  = 'TO';
KBHSMWT.SKBH_MAILADR(1)   = 'j_doe@somedomain.com';
/* First item is a normal message (text) */
KBHSMWT.SKBH_TSQ_MODE(1)  = 'TS';
KBHSMWT.SKBH_TSQUEUE(1)   = 'ENGLISH';
KBHSMWT.SKBH_ITEM_TYPE(1) = 'MAIL';
/* Second item is a graphic image (binary file) */
KBHSMWT.SKBH_TSQ_MODE(2)  = 'TS';
KBHSMWT.SKBH_TSQUEUE(2)   = 'IMBLOGO';
KBHSMWT.SKBH_ITEM_TYPE(2) = 'FILEBIN';
KBHSMWT.SKBH_MIME_TYPE(2) = 'image/jpeg';
KBHSMWT.SKBH_FILEID(2)    = 'imblogo.jpg';

/* Call the API */
EXEC CICS LINK PROGRAM('KBHSMTP')
               COMMAREA(KBHSMWT)
               LENGTH(CSTG(KBHSMWT))
               NOHANDLE;

/* Check return code */
IF EIBRESP = 0 THEN
  IF KBHSMWT.SKBH_MSGID = '' THEN
    PUT SKIP EDIT('e-mail was sent OK') (A);
  ELSE
    PUT SKIP EDIT('e-mail was not sent. Message=',
                  KBHSMWT.SKBH_MSGID) (A,A);
ELSE
  PUT SKIP EDIT('Problem during link to API. EIBRESP=',
                EIBRESP,' EIBRESP2=',EIBRESP2) (A,F(4),A,F(4));
```

Generate unique TS queue names—KBHUQNP

The unique TS queue name API is used to generate names on application TS queues, which must not collide with parallel executing programs.

The API will generate unique TS queue names. A number of counters are maintained for different prefixes. Before a name is returned, it is checked that no TS queue exists with that name.

Up to 10 TS queue names with same or different prefix can be generated in the same call.

Format

KBHUQNP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBHUQNP')
                    COMMAREA(COMMAREA)
                    NOHANDLE; ────────────────────────────────►◄

     DCL 1 COMMAREA,
          2 KBHUQWN─PTR     POINTER;
```

Parameters

Parameters as declared in PL/1:

```
/* +---------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------     | */
/* | (C) Copyright IBM Denmark. 1998.      All Rights Reserved.   | */
/* | (C) Copyright IBM Corp. 1998.         All Rights Reserved.   | */
/* |                                                               | */
/* |                                                               | */
/* +---------------------------------------------------------------+ */

/*   +-----------------------------------------------------------+   */
/*   |              IMB GENERAL INFRASTRUCTURE                   |   */
/*   |              ===============================              |   */
/*   |                                                           |   */
/*   |  STRUCTURE : KBHUQWN                                      |   */
/*   |                                                           |   */
/*   |  LENGTH    : 00200 BYTES                                  |   */
/*   |                                                           |   */
/*   |  CONTENTS  : IMB API TO GENERATE UNIQUE TS QUEUE NAME|        */
/*   |                                                           |   */
/*   |  RELATIONS : NONE.                                        |   */
/*   |                                                           |   */
/*   +-----------------------------------------------------------+   */
/*                                                                   */
/*                                                            OFFSET */
/*                                                               --- */
    3 SKBH_TSPREFIX(10)  CHAR (04),/* INP TS QUEUE PREFIX ARRAY  000 */
    3 SKBH_TSQUEUE(10)   CHAR (08),/* OUT TS QUEUE NAME ARRAY    040 */
    3 SKBH_MSGID         CHAR (10),/* MESSAGE NUMBER             120 */
    3 SKBH_RESERV        CHAR (70) /* RESERVED                   130 */
/*                                                                   */
/*                                                        TOTAL LENGTH 00200 */
/*  ==  IMB  == END OF STRUCTURE KBHUQWN ==                                  */
/*  ---------------------------------------------------------------------- */
```

SKBH_TSPREFIX
Array of 10 prefixes of TS queue names, that should be used to generate unique TS queue names.

SKBH_TSQUEUE
Array of 10 generated TS queue names.

SKBH_MSGID
Message id pointing to error message. If the message id is blank, then call was successful.

RESERVED
Future use.

Must be initialised to blanks by calling application.

Examples

Here is an example of calling the API from CSP:

```
;
;/* Call PL1 program KBHUQNP to get 3 unique TS queues
;
SET KBHUQWN EMPTY;
KBHUQWN.SKBH_TSPREFIX(1) = 'KBAK';
KBHUQWN.SKBH_TSPREFIX(2) = 'KBAK';
KBHUQWN.SKBH_TSPREFIX(3) = 'KBAL';
;
CALL KBHUQNP KBHUQWN (NOMAPS,NONCSP;
;
IF KBHUQWN.SKBH_MSGID = ' ';
  ;/* OK
  ;/*
  ;/* 3 unique TS queue names is in
  ;/* KBHUQWN.SKBH_TSQUEUE(1)   e.g. KBAK00A7
  ;/* KBHUQWN.SKBH_TSQUEUE(2)   e.g. KBAK00A8
  ;/* KBHUQWN.SKBH_TSQUEUE(3)   e.g. KBAL1AD4
  ;/*
ELSE;
  ;/* Error
  ;/* More info in KBHUQWN.SKBH_MSGID
END;
```

Processing

The TS queue KBHunqnm is used to keep the list of last used tokens for the different groups.

Allocate a VSAM data set from a pool—KBHUVSP

The VSAM pool allocate API is used to select a VSAM file from a pool of VSAM files. This is useful when a temporary VSAM file is needed for parallel executing programs, especially when interfacing with other program products which either support TS queues or VSAM files.

Normally TS queues are used as temporary storage in CICS, but if the contents would fill more than one TS queue (32K records), and only a single file can be used, it might be necessary to use other storage types.

Up to 5 VSAM files can be acquired from a pool in the same call (if the pool is big enough).

On the first call, an allocated VSAM file will be emptied and a CICS enqueue is done to prevent other tasks from getting the same VSAM file. The file can then be used in the calling program. After usage, the API is called again to free the VSAM file(s).

Format

KBHUVSP is a CICS Main program, with the following LINK syntax:

LINK Syntax (PL/I):

```
►►── EXEC CICS LINK PROGRAM('KBHUVSP')
                    COMMAREA(COMMAREA)
                    NOHANDLE; ────────────────────────────────►◄

     DCL 1 COMMAREA,
          2 KBHUVWS—PTR     POINTER;
```

Parameters

Parameters as declared in PL/1:

```
/* +----------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------      | */
/* | (C) COPYRIGHT IBM DENMARK. 1998.       ALL RIGHTS RESERVED.    | */
/* | (C) COPYRIGHT IBM CORP. 1998.          ALL RIGHTS RESERVED.    | */
/* |                                                                | */
/* |                                                                | */
/* +----------------------------------------------------------------+ */

/*   +----------------------------------------------------------+    */
/*   |              IMB GENERAL INFRASTRUCTURE                  |    */
/*   |              ==============================              |    */
/*   |                                                          |    */
/*   |  STRUCTURE : KBHUVWS                                     |    */
/*   |                                                          |    */
/*   |  LENGTH    : 00200 BYTES                                 |    */
/*   |                                                          |    */
/*   |  CONTENTS  : IMB API TO ALLOCATE VSAM FILE               |    */
/*   |                                                          |    */
/*   |  RELATIONS : NONE.                                       |    */
/*   |                                                          |    */
/*   +----------------------------------------------------------+    */
/*                                                                   */
/*                                                           OFFSET */
/*                                                              --- */
    3 SKBH_FNC          CHAR(08),/* FUNCTION CODE              000 */
    3 ISYSIDY           CHAR(04),/* APPLICATION SYSTEM ID      008 */
    3 IOPUCTY           CHAR(03),/* COUNTRY CODE               008 */
    3 SKBH_VSAM(5)      CHAR(08),/* ARRAY OF VSAM FILE NAMES   040 */
    3 SKBH_MSGID        CHAR(10),/* MESSAGE NUMBER             120 */
    3 SKBH_RESERV       CHAR(70) /* RESERVED                   130 */
/*                                                                 */
/*                                             TOTAL LENGTH 00200 */
/*  ==           == END OF STRUCTURE KBHUVWS ==                    */
/*  ------------------------------------------------------------- */
```

SKBH_FNC
: The function code to the API. Possible values: 'ALLOC','ALLOC2','ALLOC3','ALLOC4','ALLOC5' to allocate 1 to 5 VSAM files and 'FREE' to free them all again.

ISYSIDY
: Application id of calling project. This id together with the IOPUCTY forms the name of the pool.

IOPUCTY
: Country code of calling project. This value together with the ISYSIDY forms the name of the pool.

SKBH_VSAM
: Array of 5 generated VSAM file names.

SKBH_MSGID
: Message id pointing to error message. If the message id is blank, then call was successful.

SKBH_RESERV
: Future use. Has to be initiated to blanks by the calling application.

Processing

A IMB WC table entry is used to control a pool. IMB itself is using the following entry:

```
WC-Key           Cty  Value
VSAM FILES KBH   000  KBHVS 001 006 02
```

The above pool contains 6 VSAM files KBHVS001 to KBHVS006.

It will usually give the best performance, if other applications have their own pool, since taking VSAM files from the IMB pool will have a negative effect on the processing done in MailRoom.

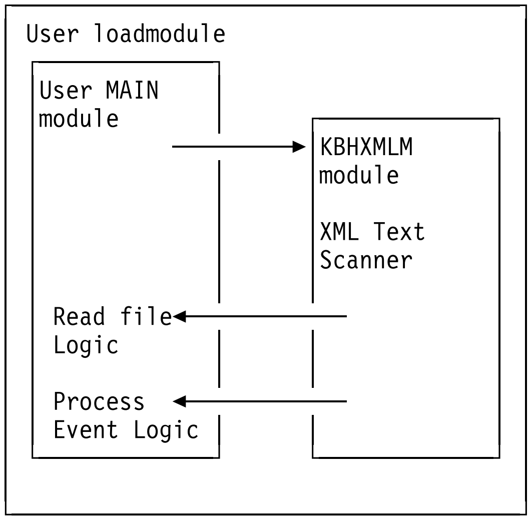
A pool is created by physically defining a number of VSAM files, for example, ZZZVS001 to ZZZVS025, create the related CICS CEDA entries and by adding an entry to the WC table (use option WCEDIT). The API should then be called with ISYSIDY: ZZZ and IOPUCTY: 000, and the entry in the WC table should be:

```
WC-Key           Cty  Value
VSAM FILES ZZZ   000  ZZZVS 001 025 02
```

XML Text Scanner, primitive XML Parser—KBHXMLM

The XML Text Scanner is a primitive XML Parser that can be used in the absence of a high performing XML Parser for CICS. This XML Text Scanner is inspired of the SAX API programming model (Simple API for XML) where a user program gets events during XML parsing.

The user program, normally a PL/I main program, defines the XML Text Scanner as an `ENTRY EXTERNAL` as well as two local procedures, one for reading the XML file and one for processing the events during XML parsing. When the XML Text Scanner is called, it will call back several times to these procedures in the main program.



The XML Text Scanner module KBHXMLM itself does not require CICS. It is therefore possible to use it outside CICS with proper implementation of the file reading in the main PL/I program.

Restrictions

The XML Text Scanner is not a real XML Parser and the following should be observed:

- XML files must be encoded in EBCDIC.
- It is a non validating parser (the DTD is not used) and only certain markup errors will be detected if the program is instructed to do so.
- Certain limits in the length of XML tag names, attribute names and attribute values exist in this program. See the structure passed to the event handler.
- Line feed and tab characters are not handled specially. Pass individual lines as separate records to the module.
- The XML Text Scanner was originally developed for CICS and PL/I and it has not been tested outside of this environment.
- The programming language of the main program should also be PL/I, but other languages might be possible to use with Language Environment.
- Always consider using a real XML Parser if one is available for your environment and programming language.

XML events

The XML Text Scanner (module KBHXMLM) will produce a number of events while the XML file is processed. It is up to the main PL/I program to use these events and the associated data.

E.g. when processing an `order` XML file, the program could capture data to be used in the order header record and process all `order_line` element groups as individual order lines. When an `order_line` tag begins, it could initiate a new order line record, then capture element data for tags inside the `order_line` tag and set related fields in the order line record. Finally when the `order_line` end tag is reached, the program could save the order line record in a database.

The following sample XML file will be used to illustrate which events are generated by the XML Text Scanner:

```
<?xml version="1.0" ?>
<!DOCTYPE test SYSTEM "test.dtd">
<!-- This is a test XML file -->
<test>
  <aaa bb="123">
    <ddd>Test XML</ddd>
    <eee ff="XYZ"/>
    <xxx>Data ln1
Data ln2
Data ln3
    </xxx>
  </aaa>
</test>
```

The following events will be generated for the above XML file:

```
Start of document
PI element:         xml                                /
PI Attribute:       version = 1.0                      /
Start of element:   test                               /test
Start of element:   aaa                                /test/aaa
Attribute:          bb = 123                           /test/aaa
Start of element:   ddd                                /test/aaa/ddd
Characters:         Test XML                           /test/aaa/ddd
End of element:     ddd                                /test/aaa/ddd
Start of element:   eee                                /test/aaa/eee
Attribute:          ff = XYZ                           /test/aaa/eee
End of element:     eee                                /test/aaa/eee
Start of element:   xxx                                /test/aaa/xxx
Characters:         Data ln1                           /test/aaa/xxx
Characters:         Data ln2                           /test/aaa/xxx
Characters:         Data ln3                           /test/aaa/xxx
End of element:     xxx                                /test/aaa/xxx
End of element:     aaa                                /test/aaa
End of element:     test                               /test
End of document
```

Format

KBHXMLM is a PL/I external procedure with the following call syntax:

Call Syntax (PL/I):

```
DCL KBHXMLM                 ENTRY EXTERNAL;
DCL KBHXML_PTR              POINTER;
DCL 1 KBHXML,
      %INCLUDE KBHXML;;
KBHXML_PTR = ADDR(KBHXML);
...

CALL KBHXMLM(KBHXML_PTR,
             MY_XML_BUF_HANDLER,
             MY_XML_DOC_HANDLER);


MY_XML_BUF_HANDLER:PROCEDURE(BUF_HNDL_PTR) REORDER;
/* local implementation of read XML file */
...
END MY_XML_BUF_HANDLER;


MY_XML_DOC_HANDLER:PROCEDURE(DOC_HNDL_PTR) REORDER;
/* local implementation of XML event handler */
...
END MY_XML_DOC_HANDLER;
```

The passed procedure names from the main PL/I program must implement the following logic:

XML_BUF_HANDLER
User procedure to perform read of the XML file (get one record at a time). When end of file is reached return a special message id.

XML_DOC_HANDLER
User procedure to receive various events during XML processing.

- Start of document
- Processing Instruction
- Processing Instruction attribute
- Start of element
- Element attribute
- Characters (element data)
- End of element
- End of document

This procedure may stop the further XML parsing with a special message id when all necessary XML elements have been found. This is useful if you are just looking for s specific tag.

Parameters

Passed structure for the XML Text Scanner. It can be found in KBH.R450.PLINCL(KBHXML)

```
/* +-------------------------------------------------------------+ */
/* |       -------- INTELLIGENT MESSAGE BROKER (IMB) ---------    | */
/* | (C) Copyright IBM Denmark. 2001.       All Rights Reserved.  | */
/* | (C) Copyright IBM Corp.    2001.       All Rights Reserved.  | */
/* |                                                             | */
/* |                                                             | */
/* +-------------------------------------------------------------+ */

/* +-------------------------------------------------------------+ */
/* | KBHXML           PROGRAM CHANGE LOG                         | */
/* | =======          ==================                         | */
/* |                                                             | */
/* | DATE     USER     COMMENT                               Rel | */
/* | -------- -------- -------------------------------------- --- | */
/* | 01/02/27 ISHLS    Initial version                       450 | */
/* |                                                             | */
/* +-------------------------------------------------------------+ */
/*                                                                 */
/*  +-----------------------------------------------------------+  */
/*  |             IMB XML TEXT SCANNER                          |  */
/*  |             ===================                           |  */
/*  |                                                           |  */
/*  |  STRUCTURE : KBHXML                                       |  */
/*  |                                                           |  */
/*  |  LENGTH    : 00200 BYTES                                  |  */
/*  |                                                           |  */
/*  |  CONTENTS  : API FOR XML TEXT SCANNER (SIMPLE XML PARSER) |  */
/*  |                                                           |  */
/*  |  RELATIONS : NONE.                                        |  */
/*  |                                                           |  */
/*  +-----------------------------------------------------------+  */
/*                                                                 */
/*                                                          OFFSET */
/*                                                             --- */
    3 INP_DEBUG_SW_ON    BIT(1),   /* DEBUG SWITCH             000 */
    3 IGNORE_XML_ERR     BIT(1),   /* IGNORE XML ERRORS        000 */
    3 SKBH_MSGID         CHAR(10), /* MESSAGE TO USER          001 */
    3 SKBH_MSGVAR        CHAR(75), /* MESSAGE VARIABLES        011 */
    3 RESERVED           CHAR(114) /* RESERVED                 086 */
/*                                                                 */
/*                                              TOTAL LENGTH 00200 */
/*  ==  END OF STRUCTURE KBHXML   ==                               */
/*  -------------------------------------------------------------- */
```

INP_DEBUG_SW_ON
Enable or disable internal debugging in KBHXMLM module.
Use `'0'B` for false and `'1'B` for true.

IGNORE_XML_ERR
Instruct KBHXMLM to ignore certain XML markup errors, e.g. unfinished tags and improper nesting of tags.
Use `'0'B` for false and `'1'B` for true.

SKBH_MSGID
: Message id with error message. If the message id is blank, then call was successful.

SKBH_MSGVAR
: Message variables for SKBH_MSGID

RESERVED
: Future use. Must be initialized to blanks by calling program.

User procedure XML_BUF_HANDLER

The XML_BUF_HANDLER procedure is the file or buffer handler. The procedure must be implemented in the main PL/I program and its entry passed to KBHXMLM. The name of the procedure does not have to be XML_BUF_HANDLER, but it should implement the functionality described here.

The procedure is used to read sequentially through an XML file and return a single record (line) each time the procedure is called. The returned record can be up to 32K in length. When end of file is reached, the procedure must return the message id *KBHXML001* to signal end of file.

Other message ids of own choice can be returned at an earlier point if errors occur. When such an error message id or end of file is returned, the XML Text Scanner will stop and return control to the main PL/I program.

The procedure should look like the following:

```
XML_BUF_HANDLER:PROCEDURE(BUF_HNDL_PTR) REORDER;
DCL BUF_HNDL_PTR            POINTER;
DCL 1 BUF_HNDL              BASED(BUF_HNDL_PTR),
     2 XML_BUF_LNG          BIN FIXED (31),
     2 XML_BUF              CHAR(32767),
     2 SKBH_MSGID           CHAR(10);

BUF_HNDL.SKBH_MSGID = ''; /* OK */

/* Read file from wherever you have it */
MY_FILE_READ
/* Check the outcome */
IF 'file read OK' THEN
DO;
  BUF_HNDL.XML_BUF     = yyy; /* Actual data   */
  BUF_HNDL.XML_BUF_LNG = xxx; /* Actual length */
END;
ELSE
  IF 'file EOF' THEN
    BUF_HNDL.SKBH_MSGID = 'KBHXML001'; /* EOF reached */
  ELSE
    BUF_HNDL.SKBH_MSGID = 'XXXYYY009'; /* Other problem */

END XML_BUF_HANDLER;
```

User procedure XML_DOC_HANDLER

The XML_DOC_HANDLER procedure is the document or event handler. The procedure must be implemented in the main PL/I program and its entry passed to KBHXMLM. The name of the procedure does not have to be XML_DOC_HANDLER, but it should implement the functionality described here.

The procedure is used to process the events that are generated by KBHXMLM during the XML parsing.

- Start of document
- Processing Instruction (PI)
- Processing Instruction attribute
- Start of element
- Element attribute
- Characters (element data)
- End of element
- End of document

This procedure can use msgid *KBHXML002* to signal soft stop (we got all elements we were looking for, skip rest of XML file). Other message ids of own choice can be returned at any time if errors occur. When such an error message id or soft stop is returned, the XML Text Scanner will stop and return control to the main PL/I program.

The procedure should look like the following:

```
XML_DOC_HANDLER:PROCEDURE(DOC_HNDL_PTR) REORDER;
DCL DOC_HNDL_PTR            POINTER;
DCL 1 DOC_HNDL              BASED(DOC_HNDL_PTR),
     2 DOC_HNDL_FNC         CHAR(02) VAR,
     2 DOC_HNDL_NAME        CHAR(100) VAR,
     2 DOC_HNDL_ATTR        CHAR(100) VAR,
     2 DOC_HNDL_PATH        CHAR(400) VAR,
     2 DOC_HNDL_DATA        CHAR(32767) VAR,
     2 SKBH_MSGID           CHAR(10);

DOC_HNDL.SKBH_MSGID = ''; /* OK */

/* Process an event */
/* Depending on the event type do whatever necessary */

SELECT (DOC_HNDL_FNC);
  WHEN('SD');           /* startDocument event           */
    /* No data available */

  WHEN('PI');           /* ProcessingInstruction event   */
    /* Name of tag in DOC_HNDL_NAME */

  WHEN('PA');           /* PIAttribute event             */
    /* Name of tag       in DOC_HNDL_NAME */
    /* Name of attribute  in DOC_HNDL_ATTR */
    /* Value of attribute in DOC_HNDL_DATA */

  WHEN('SE');           /* startElement event            */
    /* Name of tag       in DOC_HNDL_NAME */
    /* Name of path      in DOC_HNDL_PATH */

  WHEN('AT');           /* Attribute event               */
    /* Name of tag       in DOC_HNDL_NAME */
    /* Name of path      in DOC_HNDL_PATH */
    /* Name of attribute  in DOC_HNDL_ATTR */
    /* Value of attribute in DOC_HNDL_DATA */

  WHEN('CH');           /* Characters event              */
    /* Name of tag       in DOC_HNDL_NAME */
    /* Name of path      in DOC_HNDL_PATH */
    /* Element data      in DOC_HNDL_DATA */

  WHEN('EE');           /* endElement event              */
    /* Name of tag       in DOC_HNDL_NAME */
    /* Name of path      in DOC_HNDL_PATH */
    /* Element data      in DOC_HNDL_DATA (concatenated) */

  WHEN('ED');           /* endDocument event             */
    /* No data available */
  OTHERWISE;
    /* Should not occur */
END;

/* Check the outcome */
IF 'I got what I were looking for, stop scanning' THEN
  DOC_HNDL.SKBH_MSGID = 'KBHXML002'; /* Soft stop */
ELSE
  IF 'any errors' THEN
    DOC_HNDL.SKBH_MSGID = 'XXXYYY002'; /* Other problem */

END XML_DOC_HANDLER;
```

Processing

The XML Text Scanner might return one of the following message ids (field SKBH_MSGID) during the parsing of an XML file. A blank message id indicates successful completion.

KBHXML001 End of XML Document reached *(not returned)*

KBHXML002 XML processing stopped at program request *(not returned)*

KBHXML003 Incorrect level return: &1 Current level: &2 in XML line &3

KBHXML004 Invalid attribute value/format (&1) in XML line &2

KBHXML005 XML File ended before all started tags were ended

XXXYYY0nn Other error returned from file or event handler

Message id KBHXML001 (EOF) and KBHXML002 (soft stop) are not returned. Instead a blank massage id is returned indicating successful completion.

&1, &2 and &3 are message variables that should be substituted with runtime values found in field SKBH_MSGVAR.

Examples

A sample PL/I program for CICS using the XML Text Scanner can be found in KBH.R450.PLI(KBHXMPM)

This sample program can be tested from CECI as follows:

```
CECI LINK PROGRAM('KBHXMPP') COMMAREA('ttttttttn') LENGTH(200)

tttttttt is the name of a TS queue containing an XML file
         use DEMO to try with a sample XML file (internal in the pgm)
n        is a debug switch: Y or N
```

Text is written to CEEMSG during processing illustrating the events. A text message is returned in commarea upon completion.

The sample program has an event handler that prints the events as they occur and keeps track of the root element. It has two different file handlers, one that reads an XML file from a TS queue and one that reads a sample XML file from internal storage.

Chapter 17. Programming APIs and structures for CSP 3270 applications

This chapter describes of some of the IMB components that can be used in other applications and applications running under IMB control.

IMB online help system

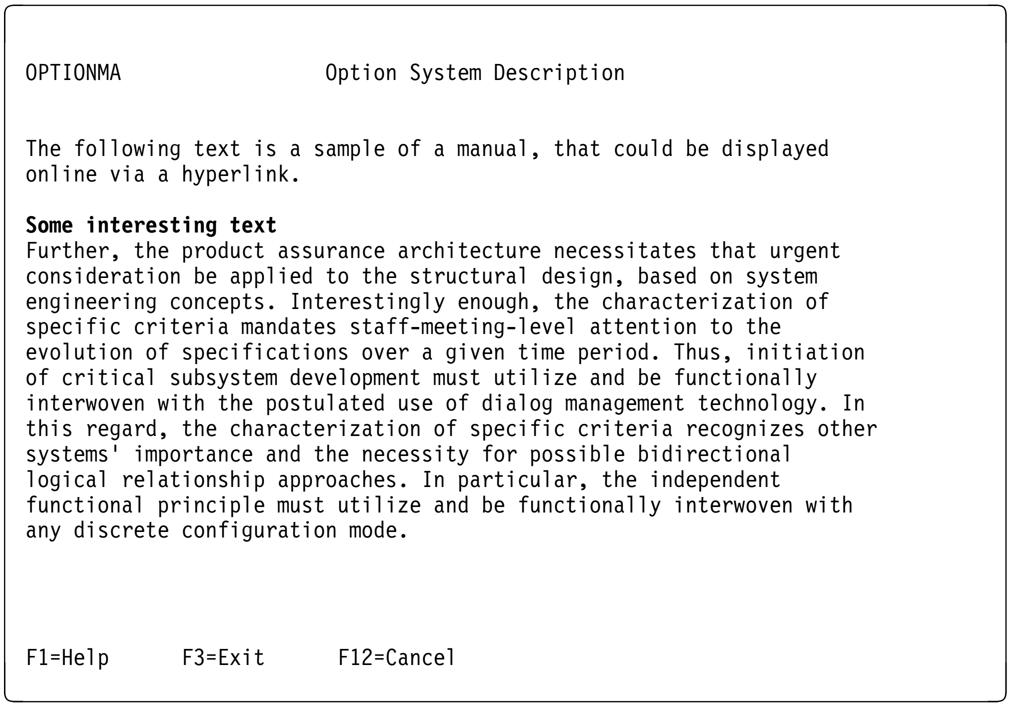
The IMB help system is a generic component to store text information and later present it to the user. It consists of a DB2 table to store the help text, a CSP Call API to present the help online, and a utility program to control the loading of new help text.

Usage

The IMB help system is activated under program control, and can display a help screen like this:

```
KBEHBMIL                      Options - Help


The panel presents a list of all options that are defined to IMB.

[See also the Option System Description]

Action codes   (Depending on authorization)
  If panel used as action list:
    D - Delete option.
    I - Insert new option.
    M - Modify option.
    C - Copy option contents to a new option.
    S - Select option for detail information.

  If panel used as prompt list:
    S - Select option for further use.
    A - Add option (and option path) to agreement set.
    R - Remove option (and all underlying options) from agreement set.

Generic Search
  Is possible by entering a search argument on the underscored lines
  on top of the list.

F1=Help      F3=Exit      F8=Forward   F12=Cancel
```

By placing the cursor on the line ***[See also the Option System Description]***, a hyperlink will be made to another help text, that could give more information about the topic. Here it would display a manual with the title *Option System Description*. All hyperlinks are displayed in yellow on the help screen.



```
OPTIONMA                 Option System Description


The following text is a sample of a manual, that could be displayed
online via a hyperlink.

Some interesting text
Further, the product assurance architecture necessitates that urgent
consideration be applied to the structural design, based on system
engineering concepts. Interestingly enough, the characterization of
specific criteria mandates staff-meeting-level attention to the
evolution of specifications over a given time period. Thus, initiation
of critical subsystem development must utilize and be functionally
interwoven with the postulated use of dialog management technology. In
this regard, the characterization of specific criteria recognizes other
systems' importance and the necessity for possible bidirectional
logical relationship approaches. In particular, the independent
functional principle must utilize and be functionally interwoven with
any discrete configuration mode.




F1=Help      F3=Exit      F12=Cancel
```

By pressing F12, the user will return to the previous panel, and F3 will return directly to the calling application.

Call Method

The help system can be called in a number of different ways, depending on what kind of help is needed.

- Panel help
- Field help
- Other help

The type of Help is specified in the SKBH_HLPTYP field and the name/Id of Help is specified in the SKBH_HLPID field.

A System ID ISYSIDY must be passed to the Help Display Module. The Language is taken directly from KAAWCOM.ZCUSRLAN, and if help in this language doesn't exist, the help system will try with the default language *UK* (English).

The following examples show how the help system can be called from CSP.

```
/*****************************************/
/* Call of help systen from CSP appl       */
/* Panel-help for active panel             */
/* CSP procedure: KBHxxP_CONV_HELP         */
/*****************************************/
;
SET KBHSHWA EMPTY;
MOVE 'XXX' TO KBHSHWA.ISYSIDY;           /* help key SYSTEM=XXX
MOVE 'PAN' TO KBHSHWA.SKBH_HLPTYP;       /* help key TYPE=PAN
MOVE KBHxxWM.MAPID TO KBHSHWA.SKBH_HLPID;/* help key ID=(mapid)
;                                        /* help key LANGUAGE=zz
CALL KBHSHAP KBHSHWA,KAAWCOM;    /* Process help request
;
```

Figure 62. Panel Help. This is the normal case, where an application wants to present some help text about the current panel. The help system should be called with a Help ID related to the current panel, for example, panelid, and a Helptype of 'PAN'.

```
/*****************************************/
/* Call of help systen from CSP appl       */
/* Mixed Field- and Panel-Help             */
/* Panel-help for active panel             */
/*****************************************/
/* CSP procedure: KBHxxP_FND_CSR           */
/*****************************************/
MOVEA ' ' TO KBHxxWM.CURSOR;
MOVE ' ' TO KBHxxWM.FLDHLPID;
;
IF KBHxxM1.FIELD01 IS CURSOR;
  KBHxxWM.CURSOR(1) = 'YES';
  KBHxxWM.FLDHLPID = 'FIELD01';   /* HelpId for this field
END;
 ...
/*****************************************/
/* CSP procedure: KBHxxP_CONV_HELP         */
/*****************************************/
;
IF KBHxxWM.FLDHLPID ¬= ' ';        /* Field help,
  SET KBHSHWA EMPTY;
  MOVE 'XXX' TO KBHSHWA.ISYSIDY;           /* help key SYSTEM=XXX
  MOVE 'FLD' TO KBHSHWA.SKBH_HLPTYP;       /* help key TYPE=PAN
  MOVE KBHxxWM.FLDHLPID TO KBHSHWA.SKBH_HLPID;/* help key ID=(mapid)
  ;                                        /* help key LANGUAGE=zz
  CALL KBHSHAP KBHSHWA,KAAWCOM;    /* Process help request
ELSE;                              /* Panel help
  SET KBHSHWA EMPTY;
  MOVE 'XXX' TO KBHSHWA.ISYSIDY;           /* help key SYSTEM=XXX
  MOVE 'PAN' TO KBHSHWA.SKBH_HLPTYP;       /* help key TYPE=PAN
  MOVE KBHxxWM.MAPID TO KBHSHWA.SKBH_HLPID;/* help key ID=(mapid)
  ;                                        /* help key LANGUAGE=zz
  CALL KBHSHAP KBHSHWA,KAAWCOM;    /* Process help request
END;
```

Figure 63. Mixed Field and Panel Help. This is an extension to panel help. Some applications might want to have help for each field on the panel as well as general help about the panel. The help system should be called with a Help ID related to the current field on the current panel, for example, PAN1FIELDXXX, and a Helptype of 'FLD'.

```
/*****************************************/
/* Call of help systen from menu         */
/* Code is placed in a stub main appl. and */
/* activated with functioncode/ZGOTO=2    */
/*****************************************/
;
IF ZGOTO = '2';
  ;
  ZGOTO = '0';
  ;
  SET KBHSHWA EMPTY;
  MOVE 'KBH' TO KBHSHWA.ISYSIDY;        /* help key SYSTEM=KBH
  MOVE 'TST' TO KBHSHWA.SKBH_HLPTYP;    /* help key TYPE=TST
  MOVE 'TEST1' TO KBHSHWA.SKBH_HLPID;   /* help key ID=TEST1
  ;                                     /* help key LANGUAGE=zz
  CALL KBHSHAP KBHSHWA,KAAWCOM;/* Process help request
  ;
  PERFORM KAAPDXF;             /* XFER/DXFR to menu or appl
END;
```

Figure 64. Other Help. Other kinds of text information can be stored and presented online. A User Manual, a Messages and Codes manual, Descriptions of WC table keys, or something else could be accessible online directly from the menu.

The help system should be called with a Help-Id related to the text, for example, XXXXXMANUAL, and a Helptype of 'MAN', 'MSG', 'WCK' or another 3 char. abbreviation.

External input

The IMB help system keeps the help text in a DB2 table (KBDVMH), and in order to make the load of new or changed text easy, a load utility program exists. The program is normally run from a batch JCL job, and takes a flat file as input. The flat file is 80 char wide.

Help text is defined as a number of text lines, that are shown together on the screen. The different parts of the text (title, body text, highlighted lines and hyperlinks) are identified by markup tags in the input file. An input file can contain a sequence of help text definitions. The text is not free format with auto-reflow. The tagged file is loaded line by line using the tags to control special functions.

Sample input file

The following input file will extract (READ) all defined help text under System KBH and having an Id stating with KBH. It will then add or replace the help text as shown in the previous screen example.

```
:HELP SYSTEM=KBH LANGUAGE=*  TYPE=*   ID=KBH*     FUNCTION=READ.
:EHELP.

:HELP SYSTEM=KBH LANGUAGE=UK TYPE=PAN ID=KBEHBMIL FUNCTION=REPLACE.
:TITLE.
Options - Help
:ETITLE.

The panel presents a list of all options that are defined to IMB.

:LINK TYPE=MAN ID=OPTIONMANUAL.
See also the &1
:ELINK.

:HP1.
Action codes   (Depending on authorization)
:EHP1.
  If panel used as action list:
    D - Delete option.
    I - Insert new option.
    M - Modify option.
    C - Copy option contents to a new option.
    S - Select option for detail information.

  If panel used as prompt list:
    S - Select option for further use.
    A - Add option (and option path) to agreement set.
    R - Remove option (and all underlying options) from agreement set.

:HP1.
Generic Search
:EHP1.
  Is possible by entering a search argument on the underscored lines
  on top of the list.

:EHELP.
.*
.*
:HELP SYSTEM=KBH LANGUAGE=UK TYPE=MAN ID=OPTIONMANUAL FUNCTION=REPLACE.
:TITLE.
Option System Description
:ETITLE.

The following text is a sample of a manual, that could be displayed
online via a hyperlink.

:HP1.
Some interesting text
:EHP1.
Further, the product assurance architecture necessitates that urgent
consideration be applied to the structural design, based on system
engineering concepts. Interestingly enough, the characterization of
specific criteria mandates staff-meeting-level attention to the
evolution of specifications over a given time period. Thus, initiation
of critical subsystem development must utilize and be functionally
interwoven with the postulated use of dialog management technology. In
this regard, the characterization of specific criteria recognizes other
systems' importance and the necessity for possible bidirectional
logical relationship approaches. In particular, the independent
functional principle must utilize and be functionally interwoven with
any discrete configuration mode.
:EHELP.
```

Sample JCL job

The JCL job KBHLMHX is a sample job to load help text into the help system. The program uses 3 datasets:

DATAIN	Contains the input file
DATAOUT	Contains the output from any function=read
DATABCK	Contains a backup of old text function=replace or delete

Syntax of input

The formal specification of the input file is given here:

help text definition

Syntax	Attributes
:help ⋮ :title ⋮ :etitle ⋮ :hp1 ⋮ :ehp1 ⋮ :link ⋮ :elink ⋮ **:ehelp**	***:HELP*** **Attributes** **Value** SYSTEM= *system id.* [LANGUAGE= {**UK**|DA|DE|ES|IT|FI|FR|NL|NO|SV}] [TYPE= {**PAN**|FLD|MAN|*other types*}] ID= *Help identification* [FUNCTION= {**REPLACE**|READ|DELETE}] [.] ***:EHELP***[.]

The different attributes have the following meaning:

SYSTEM
The project or system, this help text is related to (maximum 4 characters).

LANGUAGE
The language of the help text. Default is UK (length 2 characters).

TYPE Type of help text. Default is PAN (Panel help), other possible types are FLD (Field Help), MSG (Message Help), WCK (WC Table Keys Help), MAN (Manuals and User Guides) and other abbreviations. (length 3 char.)

ID Identification of this help text, for example panel ID or name of a manual (length maximum 16 char.)

FUNCTION
FUNCTION determines what should happen. Default is REPLACE, that will add or replace existing help text. Other functions are READ, that can extract text from the help system, and DELETE, that will remove help text. READ and DELETE can only have the :HELP and :EHELP. tags.

Title definition

Syntax	**Attributes**
:help ⋮ **:title** ⋮ **:etitle** ⋮ :ehelp	***:TITLE***[.] *title line* ***:ETITLE***[.]

There can be only one title line. (maximum length 58 char.)

Text definition

Syntax	**Attributes**
:help ⋮ :ehelp	[*text lines*]

There can be one or more text lines. (maximum length 72 char.)

Highlighted text definition

Syntax	**Attributes**
:help ⋮ **:hp1** ⋮ **:ehp1** ⋮ :ehelp	***:HP1***[.] [*Highlighted text lines*] ***:EHP1***[.]

There can be one or more text lines (maximum length 72 char.) between a set of :HP1. and :EHP1. tags. All the lines will then be highlighted.

hyperlink definition

Syntax	Attributes
:help ⋮ **:link** ⋮ **:elink** ⋮ :ehelp	***:LINK*** **Attributes** **Value** [SYSTEM= *link system id.*] [TYPE= *link type*] ID= *link help identification* [.] [*reference text to &1 hyperlink*] ***:ELINK***[.]

The different attributes has the following meaning:

SYSTEM The project or system, that the hyperlink should go to. Default is the same system as the current help text.

TYPE The type of help text, that the hyperlink should go to. Default is the same type as the current help text.

ID The identification of the help text, that the hyperlink should go to.

The following text [*reference text to &1 hyperlink*] is the text, that will appear on screen. A &1 will insert the title of the referenced help text.

Application data in KAAWCOM

When an application returns the control to IMB all allocated storage is released. If the user at a later time (same day without logging off) enters the application, the application does not know anything about the previous entered information, and the user would have to enter it again. If the application wants to keep some information during the current session it can be kept in KAAWCOM in a special application data slot.

When the user logs off from IMB, or the user is timed out due to inactivity all fields will be reset and application data in KAAWCOM are lost.

An example of application data in KAAWCOM could be a Dealer application, where an user can work with a limited set of customers. A selected customer number could be kept during the session for later use in the same or related applications. All the application has to do, is to allocate a slot in KAAWCOM, give the slot a name, place the customer number in the slot, and refer to this slot on the next invocation.

- 10 slots are available for applications.
- Application can *allocate* a slot of 200 bytes.
- A slot is identified by an unique ID, which identifies the owner (and the format) of the slot. The ID is 8 bytes long.
- The application data is kept during IMB session
- The slot might be taken by another application

Locate a previous or a new slot

When an application is started, it should either find the previous allocated application data slot or allocate a new one.

```
; /* TEST-ID is the Id of our slot
; /* MY_200_CHAR_DATA is a local structure of the slot
;
IF 'TEST-ID' IN ZAPPLDATA_ID;    /* OK, we have a slot
  ;
  MOVE ZAPPLDATA_REC(EZETST) TO MY_200_CHAR_DATA;
  ;
ELSE;                            /* No, we don't have a slot
  ;
  IF ' ' IN ZAPPLDATA_ID;        /* Any free slots ?
    ;
    MOVE 'TEST-ID' TO ZAPPLDATA_ID(EZETST);
    MOVE "Reserve" TO ZAPPLDATA_REC(EZETST);
    MOVE "My data" TO MY_200_CHAR_DATA;
    ;
  ELSE;                          /* No free slots
    ;
    ; /* We have to steal one ...
    ; /* Pickup a number (and hope nobody is hurt)
    ; /* or decide not to take a slot
    ;
    MOVE 'TEST-ID' TO ZAPPLDATA_ID(4);
    MOVE "Reserve" TO ZAPPLDATA_REC(4);
    MOVE "My data" TO MY_200_CHAR_DATA;
    ;
  END;
END;
;
```

Write data to application slot

On exit the application should save its data in the previous allocated application data.

```
IF 'TEST-ID' IN ZAPPLDATA_ID;    /* We should have a slot now
  ;
  MOVE MY_200_CHAR_DATA TO ZAPPLDATA_REC(EZETST);
  ;
END;
;
```

Common work area—KAAWCOM

KAAWCOM is the main communication work area, used by the infrastructure for building menus and navigation purposes. KAAWCOM is always passed between flexible menu (KBHMEAP) and the F/E applications. KAAWCOM has been added the area *ZAPPLDATA* where the F/E applications can store whatever data they want to. The infrastructure will not reset this part of KAAWCOM. ZAPPLDATA has an occurrence of 10, each with a length of 200 bytes and with an 8 character key to hold an application ID. All fields are initiated by application KBHINAP.

Format

Structure of KAAWCOM:

```
NAME                        LEVEL OCCURS TYPE LENGTH DESCRIPTION
Z_ALL_KAAWCOM_STRUCT         05   00001  CHA  04000
 FIELDP0                     10   00001  CHA  00013  Structure kaawip0
   QDEXLEN                   20   00001  BIN  00004  Length of interface
   IDEXSET                   20   00001  BIN  00004  Id of interface set
   CPGMSTC                   20   00001  BIN  00004  Environment status
   IPGMEWM                   20   00001  CHA  00007  Message no
 Z_USER_AND_CUSTOMER_PROFILE 10   00001  CHA  00332  User/company ref data
  ZCUSTPROF                  15   00001  CHA  00066
   ZIOPUCTY                  20   00001  CHA  00003  Country
   ZICUSPRM                  20   00001  CHA  00009  Primary cust.no.
   ZCUSTNAM                  20   00001  CHA  00030  Customer name
   ZICUSIDY                  20   00001  BIN  00009  Address id - (CS-DB)
   *                         20   00001  CHA  00020
  ZREFPROF                   15   00001  CHA  00066  Reference data
  ZSIM_CUST                  15   00001  CHA  00001  Primary acc. to simulate
  ZUSERPROF                  15   00001  CHA  00149  User ref. data
   ZUSERID                   20   00001  CHA  00008  Racf user ID
   ZCUSRLAN                  20   00001  CHA  00002  Language identifier
   ZCUSRSEC                  20   00001  CHA  00001  Security indicator
   ZMSGLANG                  20   00001  CHA  00002  Message language identifier
   ZCUSRDLM                  20   00001  CHA  00001  User delimiter in expert mode
   ZIKAAPRF                  20   00001  CHA  00008  User profile id
   ZSTDPRF                   20   00001  CHA  00008  User standard profile
   ZADMPRF                   20   00001  CHA  00008  User administration profile
   ZFUSRFID                  20   00001  CHA  00001  Show PF-keys
   ZCUSRPOC                  20   00001  CHA  00001  Panel-id option code
   ZIUSRPRM                  20   00001  CHA  00007  Customer to be simulated
   ZCUSRIAS                  20   00001  CHA  00003  User classification default
   ZCUSRCTG                  20   00001  CHA  00001  User category
   ZCUSRCSR                  20   00001  CHA  00001  Cursor default
   ZCUSRLVL                  20   00001  CHA  00003  User level
   ZIUSRHLN                  20   00001  CHA  00008  Node id - home location
   ZIUSRHLT                  20   00001  CHA  00008  Term id - home location
   ZIUSRHLP                  20   00001  CHA  00008  Printer id - home location
   ZIUSRIDY_DIAL             20   00001  CHA  00009  Dial IBM user ID
   ZIUSRIDY_SAC              20   00001  CHA  00009  SAC user ID
   ZIUSRIDY_ENG              20   00001  CHA  00009  Engine user ID
   ZIUSRIDY_IMS              20   00001  CHA  00009  IMS user ID
   ZNCUSDEPT                 20   00001  CHA  00030  Department
   ZIPRSIDY                  20   00001  BIN  00009  PersonID (Key in person table)
  *                          15   00001  CHA  00050
 Z_ENVIRONMENT               10   00001  CHA  00455  Environment ref.data
   ZNODE                     20   00001  CHA  00008  Node
   ZINITALF                  20   00001  CHA  00007  Called alf name
    ZALF                     30   00001  CHA  00001  First pos in alf-file
    *                        30   00001  CHA  00006
   ZINITAPP                  20   00001  CHA  00007  Called application name
   ZCHOICE                   20   00001  CHA  00012  COMSEC Choice number
   ZCMD                      20   00001  CHA  00057  Command line
    ZCOMMAND                 30   00001  CHA  00008  Left part of command line
    *                        30   00001  CHA  00049
   ZAPPLNAM                  20   00001  CHA  00008  Name of current application
   ZNEWAPPL                  20   00001  CHA  00008  Name of new application
   ZTRXNAM                   20   00001  CHA  00004  Current CICS transaction
   ZNEWTRX                   20   00001  CHA  00004  New CICS transaction
   ZPANELID                  20   00001  CHA  00008  Current panel
   ZOPTION                   20   00001  CHA  00006  Identify screen for return
   ZGOTO                     20   00001  CHA  00002  Information - where to go
   ZGOTOLVL                  20   00001  NUM  00001  Level in goto zstack
   ZGSTACK                   20   00015  CHA  00002  Stack
   ZGO                       20   00001  CHA  00002  Indentifier if data are passed
   ZDATA                     20   00001  CHA  00240  Data passed - total
    ZFIELDS                  30   00006  CHA  00040  Data passed
   ZCHK_MSG                  20   00001  CHA  00001
   *                         20   00001  CHA  00050
 Z_MESSAGE_HANDLING          10   00001  CHA  00359
   ZMSGNO                    20   00001  CHA  00006  Message number
   ZMSGSEVR                  20   00001  CHA  00001  Message severity
   ZMSG                      20   00001  CHA  00077  Message text
    ZMSGAMP1                 30   00001  CHA  00027  Text insertable in ZMSG (&1)
    ZMSGAMP2                 30   00001  CHA  00025  Text insertable in ZMSG (&2)
    ZMSGAMP3                 30   00001  CHA  00025  Text insertable in ZMSG (&3)
   ZSQLCOD                   20   00001  NUM  00004  SQL feedback code
   ZSQLERR                   20   00001  CHA  00077  SQL error text
   Z_LASTMSG                 20   00001  CHA  00080  Info about last message
   ZMSGID_LONG               20   00001  CHA  00010  Long message id, special use
   *                         20   00001  CHA  00104
 Z_OTHER_FIELDS              10   00001  CHA  00151
   ZDEBUG                    20   00001  CHA  00001  DEBUG: Y / N switch for test
   ZMON_FNC                  20   00001  CHA  00003  MONITOR: Function code
   ZMON_DATA                 20   00001  CHA  00128  MONITOR: Userfield data
   ZCURSOR                   20   00001  CHA  00008  Name of cursor field
   ZFOUND                    20   00001  CHA  00005  Record found indicator
   ZI                        20   00001  NUM  00003  Used for index m.m.
   ZJ                        20   00001  NUM  00003  Used for index m.m.
 *                           10   00001  CHA  00610
 ZAPPLDATA                   10   00010  CHA  00208  Application data area
   ZAPPLDATA_ID              20   00001  CHA  00008  Id of owner of appl.data recd.
   ZAPPLDATA_REC             20   00001  CHA  00200  Application def. recd. struct.
```

Processing

The F/E application can *reserve* one (or more if needed) of the 10 ZAPPLDATA areas by entering the application id in the ZAPPLDATA_ID field, and the application data in the ZAPPLDATA_rec field. The F/E applications using the application data area should first check if the ZAPPLDATA_ID 1 through 10 are filled in.

When an empty ZAPPLDATA_ID is found, the application can fill in it's own application-id and the application data in the ZAPPLDATA_REC. Now the application has stored its data in KAAWCOM, and the infrastructure will not touch the area. Whenever the F/E application needs the application data it should check the appl-id in the ZAPPLDATA_ID to ensure that no other application has overwritten the area.

All applications should in common interest only put data in an empty occurrence of ZAPPLDATA_ID/ZAPPLDATA_REC. Only in cases where all 10 occurrences are used, it is necessary to overwrite another applications area.

The F/E applications can retrieve the user country code from KAAWCOM.ZIOPUCTY for MCO operation purposes.

F-keys string builder—KBHPFKP

Given a string containing texts for all 12 F-keys and an array with marks for required F-keys, this module builds a string of only the valid F-keys (with text) to be shown on map.

The module should be called at application initialization, or prior to screen converses, if the valid F-keys change dynamically (F8 is only valid when more data is available).

Format

The module is available in a CICS version.

Call Syntax (CSP)

```
►►──CALL KBHPFKP KBHPFW (NOMAPS,NONCSP;──────────────────────►◄
```

```
CSP Working storage KBHPFW:

NAME     LEVEL OCCURS TYPE LNG  DESCRIPTION

WPFSEL    10       1  CHA   12  key to PF keys 'Y Y   YY'
  WPFSELX 15      12  CHA    1  show pf-key(x)
WPFKEYS   10       1  CHA  144  all pfkeys
  WPFKEYA 15      12  CHA   12  all pfkeys - substructure
WPFTEXT   10       2  CHA   78  selected pfkeys
```

Before the call, `WPFSELX` should be filled with non-blank character to toggle the display of the corresponding F-key.

Here is an example:

```
WPFSELX(3) = Y would cause F3 to be displayed.
```

`WPFKEYS` is filled with the total F-key string (usually extracted from a CSP table).

After a call to the module `WPFTEXT` would contain two lines of formatted F-keys.

Processing

Build a string of valid F-keys based on a string with all possible texts and an array with the currently active F-keys. The module places the text at fixed positions to avoid flipping left-right. This principle is conflicting with the CUA Architecture, which recommends two blanks between each F-key.

Examples

How to use KBHPFKP in a CSP application:

```
;
SET KBHPFW EMPTY;       /* Always initialize WS
;
RETR
  ZCUSRLAN              /* Current language
  KBHxxTx.CUSRLAN       /* Language column in table
  KBHPFW.WPFKEYS        /* Destination area
  PF;                   /* Wanted column from table
;
MOVE 'Y' TO KBHPFW.WPFSELX(1);   /* F1  valid
MOVE 'Y' TO KBHPFW.WPFSELX(3);   /* F3  valid
MOVE 'Y' TO KBHPFW.WPFSELX(12);  /* F12 valid
;
IF MORE_DATA_UPWARDS = 'Y';
  MOVE 'Y' TO KBHPFW.WPFSELX(7); /* F7  valid
END;
;
IF MORE_DATA_DOWNWARDS = 'Y';
  MOVE 'Y' TO KBHPFW.WPFSELX(8); /* F8  valid
END;
;
CALL KBHPFKP KBHPFW (NOMAPS,NONCSP;  /* Call module
;
; /* Move 2 occurences of formatted text to map
MOVEA KBHPFW.WPFTEXT TO KBHxxMxx.PF;
;
----- CONVERSE KBHxxMxx -----------
;
```

Extended scope API—KBHECAP

To test which rights for the user.

Format

The module is a CSP application with the following CALL syntax:

Call Syntax (CSP)

```
►►──CALL KBHECAP KBHECW (NOMAPS;───────────────────────────────────►◄
```

Parameters

NAME	LEV	OCC	TYPE	LEN	Inp/ Outp	Mandatory	DESC.
SKBH_FNC	10	00001	C	00008	I	Y	Function name
QROWCNT	10	00001	N	00002	O		number of lines/rows
APPLNAM	10	00001	C	00008	I	Y	Application name
USERID	10	00001	C	00008	I	Y	
MSGNO	10	00001	C	00006	O		message number
MSG_SUBST1	10	00001	C	00025	O		
MSG_SUBST2	10	00001	C	00025	O		
MSG_SUBST3	10	00001	C	00025	O		
MSG_TXT	10	00001	C	00080	O		Message txt
KBHECI_RTNCODE	10	00001	C	00001	O		Return Code
MORE	10	00001	C	00001	O		more rows ?
IUSRIDY	10	00001	C	00008	I	Y	IMB user ID
IKAAPRF	10	00001	C	00008	I		Agreement Subset ID
IKAAOPT	10	00001	C	00006	I	Y	Option id. from table
ISCOIDY	10	00001	C	00016	I	Y	Scope key
ISCOVAL	10	00001	C	00024	I	Y/N	Scope value
ISCOVAL_X	20	00024	C	00001	I		Scope value
CSCOPRI	10	00001	C	00004	I	N	PRIVILEGE
IOPUCTY	10	00001	C	00003	I	Y	Country number
ICUSPRM	10	00001	C	00009	I	Y	Primary Customer no.
SCOPE_VALUES	10	00050	C	00024	O		
SCOPE_RIGHTS	05	00050	C	00004	O		
SCRLLAREA	05	00001	C	00160	I/O		Area for scroll keys
SCRLLKEY	10	00002	C	00080	I/O		Scroll key

SKBH_FNC
The function to be done. Valid input values are:

LIST List all rights the user has.

LISTF More than 50 rows were returned by first LIST call. Second and subsequent LIST calls should use this function, and pass the returned SCRLLAREA to the API.

CHECK Check if user has the right to perform an action.

QROWCNT
How many rows (of SCOPE_VALUES) will be returned.

APPLNAM
: Application name (used by error logging).

USERID
: User Identification (used by error logging).

MSGNO
: Message number. Valid values for MSGNO are:

KBH080	Unknown function parameter passed to KBHECAP.
KBH081	No user ID, option, or scope key passed to KBHECAP.
KBH082	Option not defined.
KBH083	Scope not defined.
KBH084	No user ID or applnam passed to KBHECAP.
KBH085	User has no access to option.
KBH086	User has no access to scope.
KBH087	Trading Partner has no access to option.
KBH088	Trading Partner has no access to scope.

KBHESI_RTNCODE
: Return code, can have one of these values:

0	All Ok
1	No access, see message number or message text for further description.
5	Other error.

MORE
: If MORE = Y, the user has more than 50 scope values listed and the API should be called again in order to get the rest of the rows.

IUSRIDY
: IMB user ID.

IKAAPRF
: Not used.

IKAAOPT
: Which option has been chosen.

ISCOIDY
: Scope Key.

ISCOVAL
: Scope Value. Mandatory when SKBH_FNC = CHECK.

CSCOPRI
: Privilege should be filled out if the scope has privileges.

IOPUCTY
: Country code the user belongs to.

ICUSPRM
: Trading Partner the user belongs to.

SCOPE_VALUES
An array with all the scope values listed that user has access to. This field is always completed, regardless of which function has been chosen, if the user has any rights to use any scope.

SCOPE_RIGHTS
An array with all privileges listed that user has access to. This field is always completed, regardless of which function has been chosen, if the user has any rights to use any scope.

Description

A scope is a restriction filter telling specifically which data an user can operate on. Assume a company had 3 departments: A, B and C.

Employees of department A, should only be able to operate on data belonging to department A, with update privileges.

Employees of department B, should only be able to operate on data belonging to department B, with update privileges.

Employees of department C, should be able to operate on data belonging to department C (with update privileges), but should also be able to select(view) data from department A and B.

Finally, superior employees should have rights to operate on all data with update privileges.

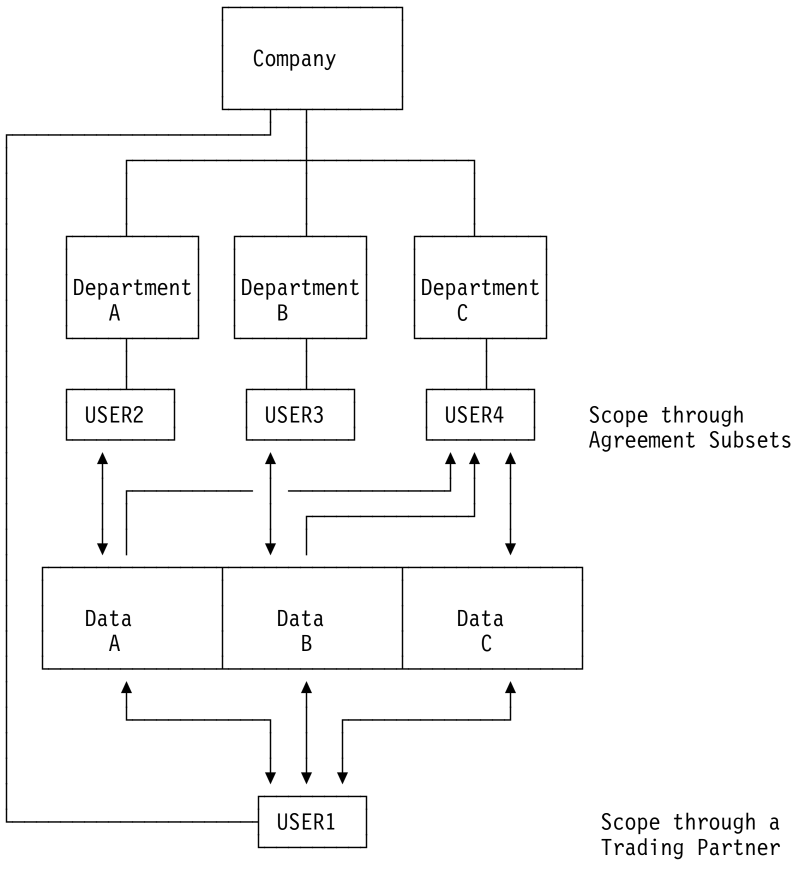


Figure 65. Example of use of the Scope facility.

Once the registrations are made in IMB a Business Application could call the API asking if the user is allowed to perform the task the user is attempting. If USER2, for example tries to select data from department B, the API should be called and a return code would tell the Business Application that the user has no such rights.

Processing

Module has two main functions:

LIST A list will be returned with all the scopes user has access to.

LISTF function is used for repeated LIST calls.

CHECK A list will be returned with all the scopes user has access to, and a return code will tell whether the user has access to the given scope/option.

For both functions the program tests whether the user has scope rights through an agreement subset or if she has scope rights through Trading Partner and if she has access to the given option.

String handler—KBHSTRP

This assembler program is for CSP to assist the work with text strings, especially with reference to IMB.

Format

The module is available in a CICS version only.

Call Syntax (CSP)

```
►►──CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;──────────────────────►◄
```

```
CSP Working storage KAAWZSTR holding data contents:

NAME           LEVEL OCCURS TYPE LNG     DESCRIPTION

ROUTINE        10    00001  CHA     8    routine name
SUBPARM        10    00001  CHA    16    Sub-parameters
TEXT1          10    00001  CHA   256    Text string 1
TEXT2          10    00001  CHA   256    Text string 2
TEXT3          10    00001  CHA   256    Text string 3
LENGHT         10    00001  BIN     4    Length of result
RESULT         10    00001  CHA   256    Result string
RNCODE         10    00001  CHA     1    Return code
```

Note: Mispelling of *LENGHT* parameter.

Processing

The program is designed as a routine with six subroutines, of which one is specified in the 'Routine name' field. The possibilities are:

1. ARRANGE
2. FOLD
3. MESSAGE
4. SUBSTR
5. TRANSLAT
6. USCORE

The load module name of the string handler is KBHSTRP, but it is also available under aliases P455A001 and P455A002 for backward compatibility.

Usage with ARRANGE

In the sub-parameter field five different parameters can be used: **,1,2,3,B.* In the three text fields three different texts can be written, which are referred to by sub-parameters *1, 2* and *3.* A text in one of these fields is read 'from first non-blank to last non-blank character' (including both characters).

A *'*'* in the sub-parameter field means a collection of blanks, which are dependent on the total number of characters, the total number of non-blanks and on the number of asterisks ('*') in the sub-parameter field. See the example later in this text. A *'B'* in the sub-parameter field means a single blank.
In the length field the maximum text length is written.
In the result field the output text is fetched.

Message	Explanation
0	Everything OK
6	The sub-parameters and written texts exceed the specified length. The result field remains unchanged.
7	In the sub-parameter field reference is made to a text, which only contains blanks. The result field is unchanged.
8	Sub-parameter field cannot be read. Result field is unchanged.
9	The specified text length is either less than 1 or greater than 256. The result field is unchanged.
A	Routine name field is incorrect. Only uppercase letters are accepted. Result field is unchanged.

Examples

How to use the ARRANGE function from a CSP application:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for ARRANGE
;
MOVE 'ARRANGE' TO KAAWZSTR.ROUTINE;
MOVE '1*2*3' TO KAAWZSTR.SUBPARM;
MOVE "abcd" TO KAAWZSTR.TEXT1;
MOVE "123" TO KAAWZSTR.TEXT2;
MOVE "abcd" TO KAAWZSTR.TEXT3;
MOVE 20 TO KAAWZSTR.LENGHT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Return code = 0
Result      = abcdbbbb123bbbbbabcd
```

The ƀ's will be represented as blanks. The same output could have been obtained if the sub-parameter field had contained 1*2*1. With an odd number of blanks an extra blank is added for each * from the right to the left.

Usage with FOLD

In the text1 field the text that is to be folded is written. In the result field the 'folded' text is fetched. All Danish lowercase letters are folded into Danish uppercase letters, while all other characters are unchanged.

Message	Explanation
0	Everything OK
A	Routine name field is incorrect. Only uppercase letters are accepted. Result field remains unchanged.

Examples

How to use the FOLD function from a CSP application:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for FOLD
;
MOVE 'FOLD' TO KAAWZSTR.ROUTINE;
MOVE "KBHSTRP can fold æ ø å to uppercase" TO KAAWZSTR.TEXT1;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Returncode = 0
Result     = KBHSTRP CAN FOLD Æ Ø Å TO UPPERCASE
```

Usage with MESSAGE

In the three text fields three different texts can be written. A text in one of these fields is read 'from first non-blank to last non-blank character' (including both characters).

In the length field the current text length is specified.

In the result field the message text from the message table is put. In this text &1, &2 and &3 will be replaced by text1, text2 and text3. When the routine is completed, the new text is written in the result field.

Message	Explanation
0	Everything OK
7	The new text will exceed the specified length. Result field remains unchanged.
8	The old text in the result field exceeds the specified length. Result field is unchanged.
9	The specified text length is either less than 1 or greater than 256. Result field is unchanged.
A	Routine name field is incorrect. Only uppercase letters are accepted.

Examples

How to use the MESSAGE function from a CSP application:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for MESSAGE
;
MOVE 'MESSAGE' TO KAAWZSTR.ROUTINE;
MOVE '99999' TO KAAWZSTR.TEXT1;
MOVE 'IBM' TO KAAWZSTR.TEXT2;
MOVE 72 TO KAAWZSTR.LENGHT;
MOVE "Employee no.: (&1 at &2) does not exist" TO KAAWZSTR.RESULT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Returncode = 0
Result     = Employee no.: (99999 at IBM) does not exist
```

Usage with SUBSTR

In the sub-parameter field (position,number) is specified.
In the text1 field is put the text string from which the sub-string is to be read. Blanks are also included.
In the length field the current text length can be specified, and it will be used if valid (1 <= length <= 256). Otherwise, default length is 256.
When the routine is completed, the sub-string will be in the result field, and the length field contains the relative position of the last character in the text1 field.

Message	Explanation
0	Everything OK
8	A sub-string which will exceed the text1 field as specified by the length field was attempted. (The length will be 256 if the specified value was incorrect.) Result field contains a text string starting with the specified position followed by as many of the following characters as possible
9	The sub-parameter field is incorrect. Result field is unchanged.
A	Routine name field is incorrect. Only uppercase letters are accepted. Result field is unchanged.

Examples

How to use the SUBSTR function from a CSP application:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for SUBSTR
;
MOVE 'SUBSTR' TO KAAWZSTR.ROUTINE;
MOVE '3,06' TO KAAWZSTR.SUBPARM;
MOVE "CCUserXXXXXXXX" TO KAAWZSTR.TEXT1;
MOVE 0 TO KAAWZSTR.LENGHT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Returncode = 0
Result     = UserXX
Length     = 14
```

Note: The same result could have been obtained if the sub-parameter field had contained 003,006 or 3,6 (but at most with three digits per input).

Usage with TRANSLAT

The TRANSLAT function is the same as the TRANSLATE function known from PL/I. The sub-parameter field is not used. Text1 is the string to be searched for possible translation of its characters. Text2 is the character expression containing the translation values of characters. Text3 is the character expression containing the characters that are to be translated. Length is the length of Text1. Result would contain the resulting string after a successful call.

Message	Explanation
0	Everything OK
9	The specified text length is either less than 1 or greater than 256. The result field is unchanged.
A	Routine name field is incorrect. Only uppercase letters are accepted. Result field is unchanged.

Examples

How to use the TRANSLAT function from a CSP application to change some characters in a string:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for TRANSLAT
;
MOVE 'TRANSLAT' TO KAAWZSTR.ROUTINE;
MOVE "This is a test" TO KAAWZSTR.TEXT1;  /* String to change
MOVE "z" TO KAAWZSTR.TEXT2;
MOVE "s" TO KAAWZSTR.TEXT3;               /* change all s to z
MOVE 15 TO KAAWZSTR.LENGHT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Return code = 0
Result      = Thiz iz a tezt
```

How to use the TRANSLAT function from a CSP application to reformat a DB2 date field:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for TRANSLAT
;
MOVE 'TRANSLAT' TO KAAWZSTR.ROUTINE;
MOVE 'GH/EF ABCD' TO KAAWZSTR.TEXT1;  /* Output mask
MOVE '1990-09-17' TO KAAWZSTR.TEXT2;  /* Input string
MOVE 'ABCD-EF-GH' TO KAAWZSTR.TEXT3;  /* Input mask
MOVE 10 TO KAAWZSTR.LENGHT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Return code = 0
Result      = 17/09 1990
```

Usage with USCORE

The USCORE function is used to set or remove underscore characters in a string to be displayed on a CSP map. The physical underscore attribute is not supported on all 3270 terminals/emulators, and as a replacement we could use the underscore character to illustrate the length of an input field. The USCORE function is able to set and remove underscore characters and left or right justify the string. The sub-parameter field contains one of the following parameters:

'1R', '1L', '1ƀ', '2R', '2L' or '2ƀ' ('ƀ' is a blank character).

- '1' gives a translation from blank to underscore.
- '2' gives a translation from underscore to blank.
- 'R' justifies the result to the right.
- 'L' justifies the result to the left.
- 'ƀ' (blank) does not justify the result.
- Text1 is the string to be processed.
- Length is the length of Text1.
- Result would contain the resulting string after a successful call.

Message	Explanation
0	Everything OK
8	Sub-parameter field cannot be read. Result field is unchanged.
9	The specified text length is either less than 1 or greater than 256. The result field is unchanged.
A	Routine name field is incorrect. Only uppercase letters are accepted. Result field is unchanged.

Examples

How to use the USCORE function from a CSP application to right-justify a string and set underscore characters before displaying on a map:

```
;
SET KAAWZSTR EMPTY;       /* Always initialize WS
;
; /* Setup parameters for USCORE
;
MOVE 'USCORE' TO KAAWZSTR.ROUTINE;
MOVE '1R' TO KAAWZSTR.SUBPARM;
MOVE "This is a test" TO KAAWZSTR.TEXT1;
MOVE 25 TO KAAWZSTR.LENGHT;
;
CALL KBHSTRP KAAWZSTR (NOMAPS,NONCSP;
;
IF KAAWZSTR.RNCODE = '0';
   /* KAAWZSTR.RESULT holds the result
END;
;
```

After the routine call:

```
Return code = 0
Result      = ___________This is a test
```

Appendix A. DB2 tables used by IMB

This appendix contains information about all DB2 tables used by IMB. The internal relations are shown in schematic form, and then the columns of each table are listed.

Understanding table relationships

A `c` on the relation line indicates that this relation is enforced by DB2 Referential Integrity with cascade delete. An `r` on the relation line indicates that this relation is enforced by DB2 Referential Integrity with delete restricted. Other relations are only enforced by program logic.

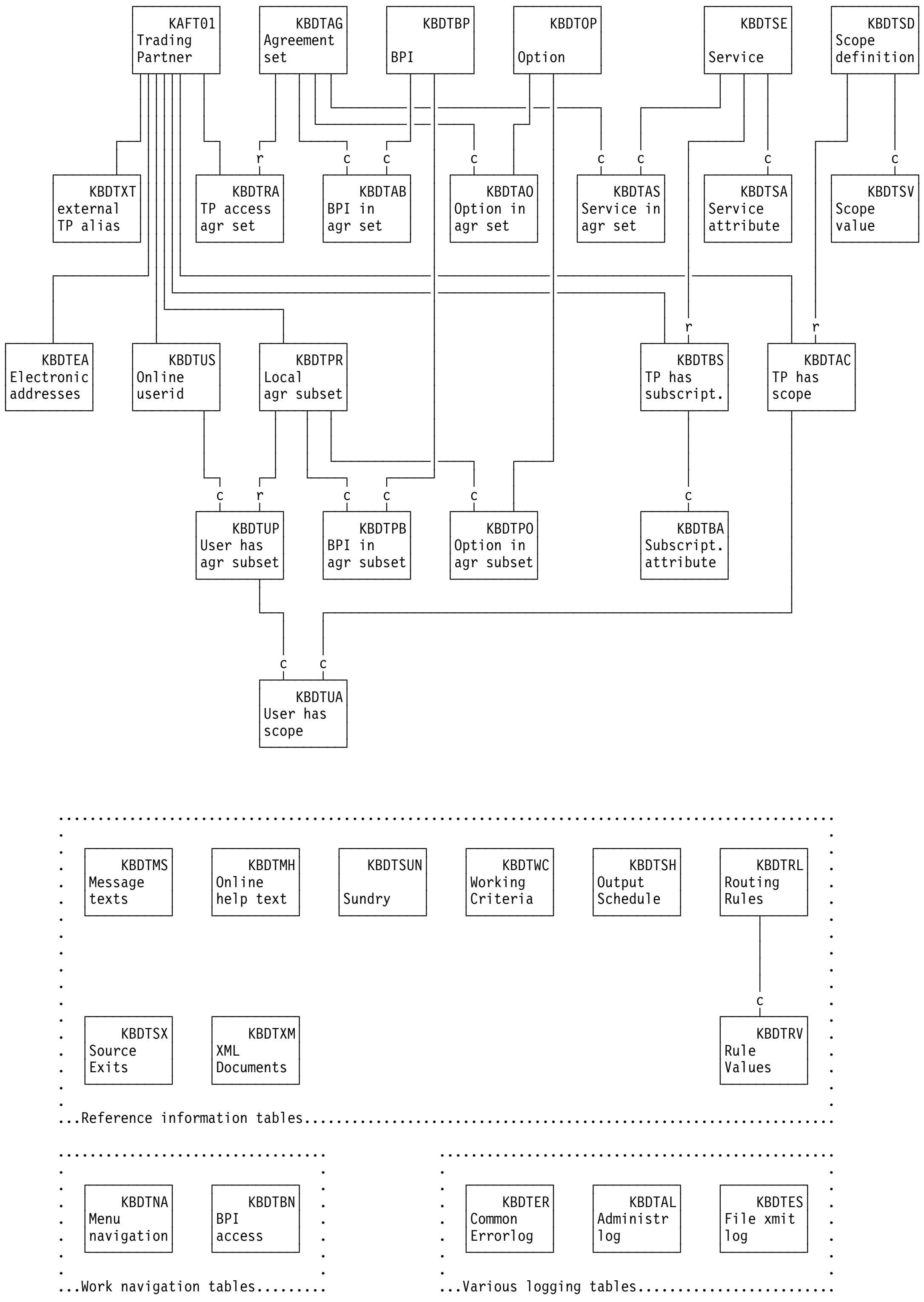


Figure 66. IMB DB2 tables for various registrations.

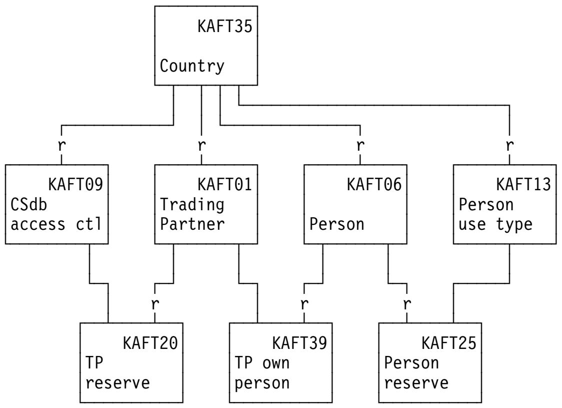


Figure 67. IMB tables for Trading Partners and persons.

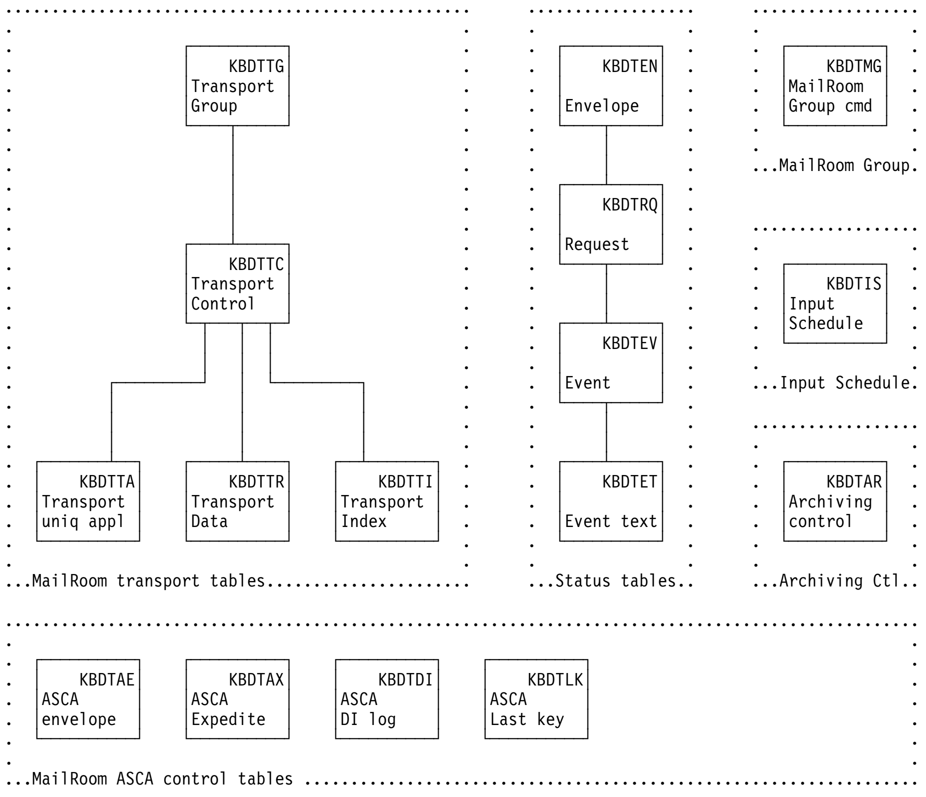


Figure 68. IMB DB2 tables used for the MailRoom.

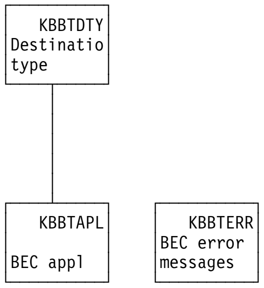


Figure 69. IMB DB2 tables used for back-end communication.

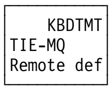


Figure 70. IMB DB2 tables used to control MailRoom TIE-MQ in remote systems.

Table descriptions

TABLE. . . . : KBDTAB
DATABASE . . : KBDD001
DESCRIPTION. : AGREEMENT SET BPI RELATIONS

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTAB	IKAAAPR	CHAR	8	AGREEMENT OPTION/SERVICE SET ID
	ISYSIDY	CHAR	4	APPLICATION SYSTEM ID
	IBPIGRP	CHAR	8	IMB BPI GROUP NAME
	IBPIFNC	CHAR	8	IMB BPI FUNCTION NAME
	IUSRUUL	CHAR	8	USERID OF LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTAC
DATABASE . . : KBDD001
DESCRIPTION. : TRADING PARTNER SCOPE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTAC	IOPUCTY	CHAR	3	COUNTRY CODE
	ICUSPRM	CHAR	9	TRADING PARTNER NUMBER
	ISCOIDY	CHAR	16	SCOPE KEY
	ISCOVAL	CHAR	24	RESTRICTION FILTER
	CSCOPRI	CHAR	4	PRIVILLIGE
	IUSRUUL	CHAR	8	USERID OF LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTAE
DATABASE . . : KBDD001
DESCRIPTION. : ASCA ENVELOPE TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTAE	SKBA_ENV_SRC	CHAR	12	MR ENVELOPE KEY
	SKBA_STATUS	CHAR	10	PROCESSING STATUS CODE FOR THE ENVELOPE
	SKBA_MSGID	CHAR	10	MESSAGENO
	SKBA_LOG_TMSTP	TIMESTMP	10	TIMESTP. WHEN LOGGED VSAM
	SKBA_TASK	CHAR	7	CICS TASK NUMBER
	SKBA_TRANS	CHAR	4	CICS TRANSACTION ID
	SKBA_APPLID	CHAR	8	CICS APPLICATION ID
	SKBA_TYPE_SRC	CHAR	8	TYPE SOURCE
	SKBA_IOPUCTY	CHAR	3	TRADING PARTNER COUNTRY
	SKBA_TPID_TO	CHAR	35	TO TRADING PARTNER (EXTERNAL)
	SKBA_TPID_FROM	CHAR	35	FROM TRADING PARTNER (EXTERNAL)
	SKBA_LAYOUT	CHAR	16	MR LAYOUT (MIXED IF MANY)
	SKBA_TOTDOC	INTEGER	4	TOTAL NO OF DOC-S IN ENVELOPE
	SKBA_GL_ACC01	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC02	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC03	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC04	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC05	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC06	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC07	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC08	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC09	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_GL_ACC10	CHAR	31	GLOBAL ACCUMULATOR
	SKBA_EF_ACCNTNO	CHAR	8	IE ACCOUNT ID - RECEIVER
	SKBA_EF_USERID	CHAR	8	IE USER ID - RECEIVER
	SKBA_EF_DESTACCT	CHAR	8	IE ACCOUNT ID - SENDER
	SKBA_EF_DESTUID	CHAR	8	IE USER ID - SENDER
	SKBA_EF_MSGSEQN	CHAR	5	IE MESSAGE SEQUENCE NUMBER
	SKBA_EF_MSGUCLS	CHAR	8	IE MESSAGE USER CLASS
	SKBA_EF_MSGDATE	CHAR	6	IE MSG DATE
	SKBA_EF_MSGTIME	CHAR	6	IE MSG TIME
	SKBA_EF_UNIQUE	CHAR	8	UNIQUEID CDH, ASSGN.BY SENDER
	SKBA_DI_EDISENDR	CHAR	35	DI ID OF EDI SENDER (LAND,KUNDE)
	SKBA_DI_EDIRECVR	CHAR	35	DI ID OF EDI RECEIVER (LAND,KUNDE)
	SKBA_DI_EDICNTLN	CHAR	14	DI INTERCHANGE CONTROL NUMBER
	SKBA_DI_CACCT	CHAR	8	IE RECEIVER ACCOUNT (POSTKASSE)
	SKBA_DI_CUSER	CHAR	8	IE RECEIVER IE USERID (POSTKASSE)
	SKBA_DI_CDACCT	CHAR	8	IE SENDER ACCOUNT (POSTKASSE)
	SKBA_DI_CDUSER	CHAR	8	IE SENDER USERID (POSTKASSE)
	SKBA_PARENT	CHAR	1	DOES REC HAVE ANY PARENTS? Y/N
	SKBA_FLOG_TMSTP	TIMESTMP	10	FINAL LOGTIMESTMP FROM IMB
	SKBA_ACK_OK	INTEGER	4	NO OF DOC-S THAT ACKNOWLEGDED OK
	SKBA_ACK_NOK	INTEGER	4	NO OF DOC-S THAT NOT ACKNOWLEGDED OK
	IUSRUUL	CHAR	8	USERID THAT UPDATED LAST TIME
	DSYSRPT	TIMESTMP	10	TIME FOR LATEST UPDATE

```
TABLE. . . . : KBDTAG
DATABASE . . : KBDD001
DESCRIPTION. : IMB AGREEMENT SET

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTAG    IKAAAPR             CHAR           8  AGREEMENT OPTION/SERVICE SET ID
          CKAAPRT             CHAR           1  AGREEMENT SET TYPE (IBM ADM ONLY)
          TKAAAPR             CHAR          40  AGREEMENT OPTION/SERVICE SET NAME
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTAL
DATABASE . . : KBDD001
DESCRIPTION. : IMB ADMINISTRATION LOG TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTAL    KBD_TABLE           CHAR           8  NAME OF TABLE AFFECTED (UPDATED)
          REQUEST             CHAR           6  INSERT/MODIFY/DELETE
          USERID              CHAR           8  USER PERFORMING REQUEST ON TABLE
          ADMLOG_DATA         CHAR         100  KEY OF ROW IN THE TABLE AFFECTED (UPDATED)
          DSYSRPT             TIMESTMP      10  WHEN DID THE TABLE UPDATE OCCUR

TABLE. . . . : KBDTAO
DATABASE . . : KBDD001
DESCRIPTION. : AGREEMENT SET OPTIONS RELATIONS

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTAO    IKAAAPR             CHAR           8  AGREEMENT OPTION/SERVICE SET ID
          IKAAOPT             CHAR           6  IMB OPTION
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTAR
DATABASE . . : KBDD001
DESCRIPTION. : ENVELOPE ARCHIVING CONTROL

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTAR    SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_TYPE_ENV       CHAR           3  TYPE OF ENVELOPE
          SKBA_ENV_SRC        CHAR          12  MR ENVELOPE KEY - SOURCE
          SKBA_DEL_TYPE       CHAR           1  A(RCHIVE) OR D(ELETE)
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE

TABLE. . . . : KBDTAS
DATABASE . . : KBDD001
DESCRIPTION. : SERVICE IN AGREEMENT SET

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTAS    IKAAAPR             CHAR           8  AGREEMENT SET NAME
          IOPUCTY             CHAR           3  COUNTRY CODE
          IPRAIDY             CHAR           8  SERVICE-ID
          IUSRUUL             CHAR           8  LATEST UPDATER USER
          DSYSRPT             TIMESTMP      10  TIMESTAMP FOR LATEST UPD.
```

```
TABLE. . . . : KBDTAX
DATABASE . . : KBDD001
DESCRIPTION. : ASCA EXPEDITE LOG TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  --------------------------------------------------------------------
KBDTAX    SKBA_CESORCIM       CHAR           6  MICROSEC REVERSED
          SKBA_TIMESTAMP      TIMESTMP      10  TIMESTAMP WHEN INSERTED
          SKBA_STATUS         CHAR          10  PROCESSING STATUS CODE
          SKBA_MSGID          CHAR          10  CURRENT MESSAGE CODE
          SKBA_LOG_TMSTP      TIMESTMP      10  TIMESTP. WHEN LOGGED VSAM (EXPEDITE)
          SKBA_TASK           CHAR           7  CICS TASK NUMBER EXPEDITE
          SKBA_TRANS          CHAR           4  CICS TRANSACTION EXPEDITE
          SKBA_APPLID         CHAR           8  CICS APPLID EXPEDITE
          SKBA_RESP           CHAR           5  COMPL. CODE FOR SEND/REC REQUESTS
          SKBA_UXTYPE         CHAR           8  TYPE OF USER EXIT
          SKBA_DIRECTION      CHAR           1  SENT(S) OR RECEIVED(R)
          SKBA_EDISQUAL       CHAR           4  ID QUALIFIER FOR EDI SENDER
          SKBA_EDISENDR       CHAR          35  ID OF EDI SENDER
          SKBA_EDIRQUAL       CHAR           4  ID QUALIFIER FOR EDI RECEIVER
          SKBA_EDIRECVR       CHAR          35  ID OF EDI RECEIVER
          SKBA_EDICNTLN       CHAR          14  INTERCHANGE CONTROL NUMBER
          SKBA_CACCT          CHAR           8  RECEIVER IE ACCOUNT
          SKBA_CUSER          CHAR           8  RECEIVER IE USERID
          SKBA_CLTYP          CHAR           1  SENDERS ALIAS TABLE TYPE
          SKBA_CLID           CHAR           3  SENDERS ALIAS TABLE ID
          SKBA_CDACCT         CHAR           8  SENDERS IE ACCOUNT
          SKBA_CDUSER         CHAR           8  SENDERS IE USERID
          SKBA_CDTYPE         CHAR           1  SENDERS DESTINATION TYPE
          SKBA_MSGUCLS        CHAR           8  MESSAGE USER CLASS                E)
          SKBA_UNIQUE         CHAR           8  UNIQUEID FROM CDH, ASSGN.BY SENDER
          SKBA_UXDATE         CHAR           8  CICS DATE IN FORMAT YYYYMMDD
          SKBA_UXTIME         CHAR           6  CICS TIME IN FORMAT TTMMSS
          SKBA_ENVSIZE        CHAR          11  ENVELOPE SIZE
          SKBA_NEDITRN        CHAR          11  NUMBER OF EDI TRANS.IN ENVELOPE
          SKBA_IESIZE         CHAR          11  IE MESSAGE SIZE, INCL.IE HEADERS
          SKBA_APPREF         CHAR          14  ASSIGNED DI APPLICATION REFERENCE
          SKBA_USERAREA       CHAR          14  WHEN SENDING,USER APPL.CAN PASS INFO
          SKBA_CONTROL        CHAR           8  1 BYTE + UNIQUE NUMERIC ID
          SKBA_TYPE_SRC       CHAR           8  SOURCE SCENARIO TYPE
          SKBA_ENV_SRC        CHAR          12  SOURCE ENVELOPE KEY
          SKBA_MATCH_DI       CHAR           1  MATCH EXPEDITE WITH DI
          IUSRUUL             CHAR           8  USERID THAT UPDATED LAST TIME
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE

TABLE. . . . : KBDTBA
DATABASE . . : KBDD001
DESCRIPTION. : SERVICE TABLE ATTRIBUTE VALUES

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTBA    IOPUCTY             CHAR           3  COUNTRY CODE
          ICUSPRM             CHAR           9  TRADING PARTNER NUMBER
          IPRAIDY             CHAR           8  SERVICE-ID
          SKBA_SUBP           CHAR           1  PART OF 1/2 SUBSCRIPTION S/D
          SKBH_ATTR           CHAR          16  ATTRIBUTE NAME
          SKBH_VALUE          CHAR          40  ATTRIBUTE VALUE

TABLE. . . . : KBDTBC
DATABASE . . : KBDD001
DESCRIPTION. : SUBSCRIPTION CONNECTION TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTBC    IOPUCTY             CHAR           3  COUNTRY CODE
          ICUSPRM_TO          CHAR           9  TO TRADING PARTNER NUMBER
          ICUSPRM_FROM        CHAR           9  FROM TRADING PARTNER NUMBER
          IPRAIDY             CHAR           8  SERVICE-ID
          SKBA_SUBP_TO        CHAR           1  PART OF 1/2 SUBSCRIPTION S
          SKBA_SUBP_FROM      CHAR           1  PART OF 1/2 SUBSCRIPTION D
          SKBH_ATTR           CHAR          16  ATTRIBUTE NAME
          SKBH_VALUE          CHAR          40  ATTRIBURE VALUE
          IUSRUUL             CHAR           8  LATEST UPDATER USER
          DSYSRPT             TIMESTMP      10  TIMESTAMP FOR LATEST UPD.
```

```
TABLE. . . . : KBDTBN
DATABASE . . : KBDD001
DESCRIPTION. : IMB BPI ACCESS LIST - NAVIGATION

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTBN    IUSRIDY            CHAR           8  IMB USERID
          ISYSIDY            CHAR           4  APPLICATION-SYSTEM ID
          IBPIGRP            CHAR           8  IMB BPI GROUP NAME
          IBPIFNC            CHAR           8  IMB BPI FUNCTION NAME
          NBPITXT            CHAR          50  IMB BPI DESCRIPTIVE TEXT
          CBPITYP            CHAR           4  BEC OR OTHER TYPE
          NBPIPGM            CHAR           8  BPI PROGRAM
          CBPISCE            CHAR           1  BPI SECURITY
          NBPIXIT            CHAR           8  BPI DATA CONVERSION EXIT
          CBPIENA            CHAR           1  BPI ENABLE-DISABLE SWITCH
          CBPIPM1            CHAR          20  BPI PARAMETER NO. 1
          CBPIPM2            CHAR          20  BPI PARAMETER NO. 2
          CBPIPM3            CHAR          20  BPI PARAMETER NO. 3

TABLE. . . . : KBDTBP
DATABASE . . : KBDD001
DESCRIPTION. : IMB BUSINESS PROCESS INTERFACE

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTBP    ISYSIDY            CHAR           4  APPLICATION-SYSTEM ID
          IBPIGRP            CHAR           8  IMB BPI GROUP NAME
          IBPIFNC            CHAR           8  IMB BPI FUNCTION NAME
          NBPITXT            CHAR          50  IMB BPI DESCRIPTIVE TEXT
          CBPITYP            CHAR           4  BEC OR OTHER TYPE
          NBPIPGM            CHAR           8  BPI PROGRAM
          CBPISCE            CHAR           1  BPI SECURITY
          NBPIXIT            CHAR           8  BPI DATA CONVERSION EXIT
          CBPIENA            CHAR           1  BPI ENABLE-DISABLE SWITCH
          CBPIPM1            CHAR          20  BPI PARAMETER NO. 1
          CBPIPM2            CHAR          20  BPI PARAMETER NO. 2
          CBPIPM3            CHAR          20  BPI PARAMETER NO. 3
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTBS
DATABASE . . : KBDD001
DESCRIPTION. : TRADING PARTNER SUBSCRIPTION TABLE

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTBS    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IPRAIDY            CHAR           8  SERVICE-ID
          SKBA_SUBP          CHAR           1  PART OF 1/2 SUBSCRIPTION S/D
          SKBA_NSUBENA       CHAR           1  STATUS (EN-/DIS-ABLED)
          IUSRUUL            CHAR           8  LATEST UPDATER USER
          DSYSRPT            TIMESTMP      10  TIMESTAMP FOR LATEST UPD.

TABLE. . . . : KBDTDI
DATABASE . . : KBDD001
DESCRIPTION. : ASCA DI LOG TABLE

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTDI    SKBA_CESORCIM      CHAR           6  MICROSEC REVERSED
          SKBA_TIMESTAMP     TIMESTMP      10  TIMESTAMP WHEN INSERTED
          SKBA_STATUS        CHAR          10  PROCESSING STATUS CODE
          SKBA_MSGID         CHAR          10  CURRENT MESSAGE CODE
          SKBA_LOG_TMSTP     TIMESTMP      10  TIMESTP. WHEN LOGGED VSAM (EXPEDITE)
          SKBA_TASK          CHAR           7  CICS TASK NUMBER EXPEDITE
          SKBA_TRANS         CHAR           4  CICS TRANSACTION EXPEDITE
          SKBA_APPLID        CHAR           8  CICS APPLID EXPEDITE
          SKBA_RESP          CHAR           5  COMPL. CODE FOR SEND/REC REQUESTS
          SKBA_EDISQUAL      CHAR           4  ID QUALIFIER FOR EDI SENDER
          SKBA_EDISENDR      CHAR          35  ID OF EDI SENDER
          SKBA_EDIRQUAL      CHAR           4  ID QUALIFIER FOR EDI RECEIVER
          SKBA_EDIRECVR      CHAR          35  ID OF EDI RECEIVER
          SKBA_EDICNTLN      CHAR          14  INTERCHANGE CONTROL NUMBER
          SKBA_DE_CESORCIM   CHAR           6  MICROSEC REVERSED    (FKEY TO KBDVDE)
          SKBA_DE_TMSTP      TIMESTMP      10  TIMESTAMP WHEN INSERTED (FKEY TO KBDVDE)
          SKBA_ENV_SRC       CHAR          12  SOURCE ENVKEY (FKEY TO KBDVAE)
          IUSRUUL            CHAR           8  USERID THAT UPDATED LAST TIME
          DSYSRPT            TIMESTMP      10  TIME FOR LATEST UPDATE
```

```
TABLE. . . . : KBDTEA
DATABASE . . : KBDD001
DESCRIPTION. : IMB ELECTRONIC ADDRESS TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ----------------------------------------------------------------------
KBDTEA    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBH_MAILKEY        INTEGER        4  ELECTRONIC ADDRESS KEY
          SKBH_MAILSCOPE      CHAR           1  USAGE SCOPE OF THIS ADDRESS
          ICUSPRM             CHAR           9  TRADING PARTNER NUMBER
          SKBH_MAILTYPE       CHAR           2  TYPE OF ELECTRONIC ADDRESS
          SKBH_MAILADR        CHAR          80  ADDRESS (LONG)
          SKBH_MAILADR1       CHAR          16  ADDRESS (SHORT) PART1 - USERID
          SKBH_MAILADR2       CHAR          16  ADDRESS (SHORT) PART2 - NODEID
          SKBH_MAILADR3       CHAR          16  ADDRESS (SHORT) PART3 - NETWORK
          SKBH_MAILADR4       CHAR          16  ADDRESS (SHORT) PART4 - OPT
          SKBH_MAILADR5       CHAR          16  ADDRESS (SHORT) PART5 - OPT
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTEN
DATABASE . . : KBDD001
DESCRIPTION. :

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTEN    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_LAYOUT         CHAR          16  FORMAT OF THE DOCUMENT
          SKBA_STATUS         CHAR          10  PROCESSING STATUS CODE FOR THE ENVELOPE
          ISYSIDY             CHAR           4  APPLICATION PROJECT ID
          IPRAIDY             CHAR           8  SERVICE-ID
          ICUSPRM_TO          CHAR           9  TO TRADING PARTNER ID
          ICUSPRM_FROM        CHAR           9  FROM TRADING PARTNER ID
          SKBA_TYPE_ENV       CHAR           3  TYPE OF ENVELOPE
          SKBA_TYPE_SRC       CHAR           8  SOURCE SCENARIO TYPE
          SKBA_TYPE_DST       CHAR           8  DESTINATION SCENARIO TYPE
          SKBA_ENV_SRC        CHAR          12  MR ENVELOPE KEY - SOURCE
          SKBA_ENV_DST        CHAR          12  MR ENVELOPE KEY - DESTINATION
          SKBA_ORIGINATOR     CHAR          80  - ASSIGNED BY SOURCE API
          SKBA_TOTDOC         INTEGER        4  TOTAL NUMBER OF DOCUMENTS IN ENVELOPE
          SKBA_SEC_OK         INTEGER        4  NO OF DOC-S WITH SEQURITY CHECK OK
          SKBA_SEC_NOK        INTEGER        4  NO OF DOC-S WITH SEQURITY CHECK NOT OK
          SKBA_PRP_OK         INTEGER        4  NO OF DOC-S THAT PREPROCESSED OK
          SKBA_PRP_NOK        INTEGER        4  NO OF DOC-S THAT NOT PREPROCESSED OK
          SKBA_SNT_OK         INTEGER        4  NO OF DOC-S THAT SENT OK
          SKBA_SNT_NOK        INTEGER        4  NO OF DOC-S THAT NOT SENT OK
          SKBA_ACK_OK         INTEGER        4  NO OF DOC-S THAT ACKNOWLEGDED OK
          SKBA_ACK_NOK        INTEGER        4  NO OF DOC-S THAT NOT ACKNOWLEGDED OK
          IUSRUUL             CHAR           8  USERID THAT UPDATED LAST TIME
          SKBA_DATE           DATE           4  DATE WHEN REQUEST WAS INSERTED
          SKBA_TMSTAMP        TIMESTMP      10  TIME WHEN REQUEST WAS INSERTED
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE

TABLE. . . . : KBDTER
DATABASE . . : KBDD001
DESCRIPTION. : COMMON ERRORLOG TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTER    SKBH_ELOG_SYS       CHAR           8  SYSTEM ID
          SKBH_ELOG_TIME      TIMESTMP      10  TIMESTAMP (ERROR TIME)
          SKBH_ELOG_LBNO      INTEGER        4  SEQ NUMBER
          SKBH_ELOG_TRX       CHAR           8  TRANSACTION
          SKBH_ELOG_TASK      CHAR           7  TASK NUMBER
          SKBH_ELOG_TERM      CHAR           8  TERMINAL ID
          SKBH_ELOG_USER      CHAR           8  RACF USERID
          SKBH_ELOG_PROG      CHAR           8  PROGRAM
          SKBH_ELOG_SUBRUT    CHAR          18  PROCESS
          SKBH_ELOG_CFROM     CHAR           8  CALLED FROM PGM
          SKBH_ELOG_OBJECT    CHAR          16  OBJECT / TABLE
          SKBH_ELOG_SUBSYS    CHAR           8  SUBSYSTEM
          SKBH_ELOG_ALERT     CHAR           1  ALERT SWITCH
          SKBH_MSGID          CHAR          10  MESSAGE ID
          SKBH_MSGSEVR        CHAR           1  SEVERITY
          SKBH_ELOG_TEXT1     CHAR          80  ERROR TEXT LINE 1
          SKBH_ELOG_TEXT2     CHAR          80  ERROR TEXT LINE 2
          SKBH_ELOG_TEXT3     CHAR          80  ERROR TEXT LINE 3
          SKBH_ELOG_TEXT4     CHAR          80  ERROR TEXT LINE 4
          SKBH_ELOG_TEXT5     CHAR          80  ERROR TEXT LINE 5
          SKBH_ELOG_TEXT6     CHAR          80  ERROR TEXT LINE 6
          SKBH_ELOG_TEXT7     CHAR          80  ERROR TEXT LINE 7
```

```
TABLE. . . . : KBDTES
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM EXPORT STATUS TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTES    KAAILUSR            CHAR           8  USERID/SERVER OWNER OF A EXPORT
          KAAILDAT            DATE           4  THE DATE OF THE EXPORT
          KAAILTOK            CHAR           4  THE TOKEN (SERIAL NO ON DAY)
          KAAILTYP            CHAR           2  EXPORT TYPE COMMUNICATION SYSTEM
          KAAILSTA            CHAR           9  CURRENT STATUS OF THE EXPORT
          KAAILSHO            CHAR           3  HAS THE MESSAGE BEEN SHOWN
          KAAILHLN            CHAR           8  DESTINATION NODE
          KAAILUSL            CHAR           8  DESTINATION USERID
          KAAILR              CHAR          13  LAST RETURN CODE
          IUSRUUL             CHAR           8  RECORD LAST UPDATE USERID
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE

TABLE. . . . : KBDTET
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM EVENT TEXT TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTET    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBA_ENVKEY         CHAR          12  ENVELOPE KEY FOR DERIVED ENVELOPE
          SKBA_DOCSEQNO       INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          SKBA_STATUS         CHAR          10  PROCESSING STATUS CODE FOR THE REQUEST
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE
          SKBA_LINNO          INTEGER        4  LINE NUMBER
          SKBA_EVTEXT         VARCHAR      255  ADDITIONAL EVENT TEXT

TABLE. . . . : KBDTEV
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM EVENT STATUS TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTEV    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBA_ENVKEY         CHAR          12  ENVELOPE KEY FOR DERIVED ENVELOPE
          SKBA_DOCSEQNO       INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          SKBA_STATUS         CHAR          10  PROCESSING STATUS CODE FOR THE REQUEST
          SKBH_MSGID          CHAR          10  MESSAGE NUMBER
          SKBA_MSG_VAR        CHAR          75  MESSAGE TEXT
          IUSRUUL             CHAR           8  USERID THAT UPDATED LAST TIME
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE

TABLE. . . . : KBDTIS
DATABASE . . : KBDD001
DESCRIPTION. : INPUT SCHEDULE CONTROL TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTIS    IOPUCTY             CHAR           3  COUNTRY CODE
          ICUSPRM             CHAR           9  TRADING PARTNER NUMBER
          IPRAIDY             CHAR           8  SERVICE ID
          SKBA_ACT_ARR        TIMESTMP      10  ACTUAL ARRIVAL
          SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_E_NEXT_ARR     TIMESTMP      10  EARLIEST NEXT ARRIVAL
          SKBA_NEXT_ALERT     TIMESTMP      10  NEXT ALERT

TABLE. . . . : KBDTLK
DATABASE . . : KBDD001
DESCRIPTION. : ASCA LAST KEY USED

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTLK    SKBA_LAST_KEY       TIMESTMP      10  KEY USED LAST TIME (TIMESTAMP)
          SKBA_LAST_KEY1      TIMESTMP      10  KEY USED LAST TIME 1ST GENERATION
          SKBA_LAST_KEY2      TIMESTMP      10  KEY USED LAST TIME 2ND GENERATION
          SKBA_LAST_KEY3      TIMESTMP      10  KEY USED LAST TIME 3TH GENERATION
          IUSRUUL             CHAR           8  USERID THAT UPDATED LAST TIME
          DSYSRPT             TIMESTMP      10  TIMESTAMP FOR LATEST UPDATE
```

TABLE. . . . : KBDTMG
DATABASE . . : KBDD001
DESCRIPTION. : IMB MAILROOM GROUP COMMAND

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTMG	IUSRUUL	CHAR	8	USERID
	SKBA_ENVKEY	CHAR	12	ENVELOPE KEY FOR DERIVED ENVELOPE
	IOPUCTY	CHAR	3	COUNTRY CODE
	SKBA_DOCSEQNO	INTEGER	4	DOCUMENT SEQUENCE NUMBER IN INVELOPE
	SKBA_LAYOUT	CHAR	16	FORMAT OF THE DOCUMENT
	SKBA_STATUS	CHAR	10	PROCESSING STATUS CODE FOR THE ENVELOPE
	ICUSPRM_TO	CHAR	9	TO TRADING PARTNER ID
	ICUSPRM_FROM	CHAR	9	FROM TRADING PARTNER ID
	SKBA_TYPE_ENV	CHAR	3	TYPE OF ENVELOPE
	SKBA_TYPE_DST	CHAR	8	DESTINATION SCENARIO TYPE
	SKBA_TMSTAMP	TIMESTMP	10	TIME WHEN REQUEST WAS INSERTED
	SKBA_PROCESSED	CHAR	10	ACTION PROCESSED
	SKBH_MSGID	CHAR	10	MESSAGE NUMBER
	SKBA_MSG_VAR	CHAR	75	MESSAGE TEXT
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF INSERT TIME

TABLE. . . . : KBDTMH
DATABASE . . : KBDD001
DESCRIPTION. : IMB HELP PANEL TEXT TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTMH	ISYSIDY	CHAR	4	IDENTIFIER APPLICATION-SYSTEM
	CUSRLAN	CHAR	2	LANGUAGE CODE
	SKBH_HLPTYP	CHAR	3	HELP TEXT TYPE (PAN,FLD,..)
	SKBH_HLPID	CHAR	16	HELP IDENTIFICATION (PANELID)
	SKBH_LINENO	SMALLINT	2	LINE NUMBER
	SKBH_LINEATT	CHAR	1	LINE ATTRIBUTES
	SKBH_HLPTEXT	CHAR	72	HELP TEXT
	IUSRUUL	CHAR	8	USERID OF LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF LATEST UPDATE

TABLE. . . . : KBDTMS
DATABASE . . : KBDD001
DESCRIPTION. : IMB MESSAGE TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTMS	SKBH_MSGID	CHAR	10	MESSAGE NUMBER
	SKBH_MSGLANG	CHAR	2	MESSAGE LANGUAGE
	SKBH_MSGSEVR	CHAR	1	MESSAGE SEVERITY
	SKBH_MSGTEXT	CHAR	80	MESSAGE TEXT

TABLE. . . . : KBDTMT
DATABASE . . : KBDD002
DESCRIPTION. : LOCAL TIE-MQ TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTMT	IOPUCTY	CHAR	3	COUNTRY CODE
	ISYSIDY	CHAR	4	TIE APPLICATION
	ITRNRCP	CHAR	10	TIE RECIPIENT
	SKBA_MQQNAME	CHAR	48	MQ QUEUE FOR MESSAGE
	SKBA_MQQERR	CHAR	48	MQ QUEUE FOR ERRORS
	SKBA_MQMNAME	CHAR	48	MQ MANAGER
	SKBA_RES1	CHAR	10	RESERVED

TABLE. . . . : KBDTNA
DATABASE . . : KBDD001
DESCRIPTION. : IMB NAVIGATOR WORK TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTNA	IUSRIDY	CHAR	8	IMB USERID
	IKAAOPT	CHAR	6	IMB OPTION
	IKAAPOP	CHAR	6	IMB NAVIGATION WORK FIELD
	NKAAMNE	CHAR	6	IMB OPTION NAME
	CKAAFNK	CHAR	2	IMB OPTION FUNCTION CODE
	NKAATXT	CHAR	50	IMB OPTION DESCRIPTIVE TEXT
	NKAAHDR	CHAR	60	IMB OPTION HEADER
	NKAAAPL	CHAR	7	IMB OPTION (CSP)APPLICATION
	NKAAALF	CHAR	7	IMB OPTION (CSP)ALF WITH NKAAAPL
	NKAATRX	CHAR	4	IMB OPTION CICS TRANSACTION
	CKAAIAS	CHAR	3	IMB OPTION CLASSIFICATION CODE
	CKAALVL	CHAR	3	IMB OPTION LEVEL
	CKAALAN	CHAR	2	IMB OPTION LANGUAGE

```
TABLE. . . . : KBDTOP
DATABASE . . : KBDD001
DESCRIPTION. : IMB OPTIONS

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTOP    IKAAOPT            CHAR           6  IMB OPTION
          NKAAMNE            CHAR           6  IMB OPTION NAME
          CKAAFNK            CHAR           2  IMB OPTION FUNCTION CODE
          NKAATXT            CHAR          50  IMB OPTION DESCRIPTIVE TEXT
          NKAAHDR            CHAR          60  IMB OPTION HEADER
          NKAAAPL            CHAR           7  IMB OPTION (CSP)APPLICATION
          NKAAALF            CHAR           7  IMB OPTION (CSP)ALF WITH NKAAAPL
          NKAATRX            CHAR           4  IMB OPTION CICS TRANSACTION
          CKAALAN            CHAR           2  IMB OPTION LANGUAGE
          CKAAIAS            CHAR           3  IMB OPTION CLASSIFICATION CODE
          CKAALVL            CHAR           3  IMB OPTION LEVEL
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTPB
DATABASE . . : KBDD001
DESCRIPTION. : BPIS IN LOCAL OPTION SET

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTPB    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IKAAPRF            CHAR           8  LOCAL OPTION SET ID
          ISYSIDY            CHAR           4  APPLICATION SYSTEM ID
          IBPIGRP            CHAR           8  IMB BPI GROUP NAME
          IBPIFNC            CHAR           8  IMB BPI FUNCTION NAME
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE

TABLE. . . . : KBDTPO
DATABASE . . : KBDD001
DESCRIPTION. : TRADING PARTNER LOCAL OPTION SET CONTENTS

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTPO    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IKAAPRF            CHAR           8  LOCAL OPTION SET ID
          IKAAOPT            CHAR           6  IMB OPTION
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (NEW FORMAT)

TABLE. . . . : KBDTPR
DATABASE . . : KBDD001
DESCRIPTION. :  TRADING PARTNER LOCAL OPTION SET HEADER

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTPR    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IKAAPRF            CHAR           8  LOCAL OPTION SET ID
          TKAAPRF            CHAR          40  LOCAL OPTION SET DESCRIPTION
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTRA
DATABASE . . : KBDD001
DESCRIPTION. : TRADING PARTNER ACCESS TO AGREEMENT SET

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  -----------------  --------  ------  ------------------------------------------------------------
KBDTRA    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IKAAAPR            CHAR           8  AGREEMENT OPTION/SERVICE SET ID
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

TABLE. . . . : KBDTRL

```
DATABASE . . : KBDD001
DESCRIPTION. : ROUTING RULE TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTRL    IOPUCTY             CHAR           3  COUNTRY CODE
          ISYSIDY             CHAR           4  APPLICATION ID
          SKBA_RULESET        CHAR          16  RULESET NAME (RULEKEY)
          SKBA_RULENO         CHAR           3  RULENUMBER WITHIN RULESET
          SKBA_RULETP         CHAR           2  RULETYPE
          SKBA_RULEDET        VARCHAR      300  RULE DETAIL
          SKBA_RULEDEST       CHAR          35  REF. TO TP OR DISTRIBUTION LIST
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

TABLE. . . . : KBDTRQ

```
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM REQUEST TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTRQ    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBA_ENVKEY         CHAR          12  ENVELOPE KEY FOR DERIVED ENVELOPE
          SKBA_DOCSEQNO       INTEGER        4  DOCUMENT SEQUENCE NUMBER IN INVELOPE
          SKBA_TMSTAMP        TIMESTMP      10  TIME WHEN REQUEST WAS INSERTED
          SKBA_STATUS         CHAR          10  PROCESSING STATUS CODE FOR THE REQUEST
          SKBA_STAT_SEC       CHAR           1  SECURITY CHECHED: BLANK,0,1 OR S
          SKBA_STAT_PRP       CHAR           1  PREPROCESSED    : BLANK,0,1 OR S
          SKBA_STAT_SNT       CHAR           1  SENT            : BLANK,0,1 OR S
          SKBA_STAT_ACK       CHAR           1  ACKNOWLEDGED    : BLANK,0,1 OR S
          SKBA_REF_DATA       CHAR          40  REFERENCE DATA
          SKBA_REF_DATA2      CHAR          40  REFERENCE DATA (FROM DEST APPL)
          SKBA_SRC_INF1       CHAR          50  SOURCE SPECIFIC INFORMATION
          SKBA_SRC_INF2       CHAR          50  SOURCE SPECIFIC INFORMATION
          SKBA_SRC_INF3       CHAR          50  SOURCE SPECIFIC INFORMATION
          SKBA_SRC_INF4       CHAR          50  SOURCE SPECIFIC INFORMATION
          SKBA_DST_INF1       CHAR          50  DESTINATION SPECIFIC INFORMATION
          SKBA_DST_INF2       CHAR          50  DESTINATION SPECIFIC INFORMATION
          SKBA_DST_INF3       CHAR          50  DESTINATION SPECIFIC INFORMATION
          SKBA_DST_INF4       CHAR          50  DESTINATION SPECIFIC INFORMATION
          SKBA_SYA_INF1       CHAR         120  SYSTEM ACK CONTROL INFO
          SKBA_PROC_CNTL      CHAR          50  INTERNAL PROCESSING CONTROL
          SKBA_ACKLVL         CHAR           1  WANTED ACKNOWLEDGMENT LEVEL
          SKBA_ACKLVL_RCH     CHAR           1  REACHED ACKNOWLEDGMENT LEVEL
          IUSRUUL             CHAR           8  USERID THAT UPDATED LAST TIME
          DSYSRPT             TIMESTMP      10  TIME FOR LATEST UPDATE
```

TABLE. . . . : KBDTRV

```
DATABASE . . : KBDD001
DESCRIPTION. : ROUTING RULE VALUESETS

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ---------------------------------------------------------------------
KBDTRV    IOPUCTY             CHAR           3  COUNTRY CODE
          ISYSIDY             CHAR           4  APPLICATION ID
          SKBA_RULESET        CHAR          16  RULESET NAME (RULEKEY)
          SKBA_RULENO         CHAR           3  RULENUMBER WITHIN RULESET
          SKBA_RULEVSET       CHAR           8  RULE VALUESET
          SKBA_RULEVAL        CHAR          75  RULE VALUE
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

TABLE. . . . : KBDTSA

```
DATABASE . . : KBDD001
DESCRIPTION. : SERVICE TABLE ATTRIBUTE VALUES

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTSA    IOPUCTY             CHAR           3  COUNTRY CODE
          IPRAIDY             CHAR           8  SERVICE-ID
          SKBH_ATTR           CHAR          16  ATTRIBUTE NAME
          SKBH_VALUE          CHAR          40  ATTRIBUTE VALUE
```

TABLE. . . . : KBDTSD
DATABASE . . : KBDD001
DESCRIPTION. : IMB SCOPE DEFINITION TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTSD	IOPUCTY	CHAR	3	COUNTRY CODE
	ISCOIDY	CHAR	16	SCOPE KEY
	NSCOTXT	CHAR	40	DESCRIPTIVE TEXT
	CSCOINV	CHAR	1	INDICATOR FOR VALIDATE AND PROMPT
	CSCOPR1	CHAR	4	PRIVILEGE 1
	CSCOPR2	CHAR	4	PRIVILEGE 2
	CSCOPR3	CHAR	4	PRIVILEGE 3
	CSCOPR4	CHAR	4	PRIVILEGE 4
	CSCOPR5	CHAR	4	PRIVILEGE 5
	CSCOINL	CHAR	2	MAX LENGTH OF SCOPE VALUE
	IUSRUUL	CHAR	8	USERID OF LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTSE
DATABASE . . : KBDD001
DESCRIPTION. : SERVICE MAIN TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTSE	IOPUCTY	CHAR	3	COUNTRY CODE
	IPRAIDY	CHAR	8	SERVICE-ID
	ISYSIDY	CHAR	4	IDENTIFIER APPLICATION-SYSTEM
	SKBA_LAYOUT	CHAR	16	FORMAT OF THE DOCUMENT
	SKBA_NPRADSC	CHAR	40	SERVICE DESCRIPTION
	SKBA_TYPE_SUB	CHAR	1	SUBSCRIPTION TYPE
	SKBA_TYPE_SRC	CHAR	8	SOURCE SCENARIO TYPE
	SKBA_TYPE_DST	CHAR	8	DESTINATION SCENARIO TYPE
	IUSRUUL	CHAR	8	USERID FOR LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIME FOR LATEST UPDATE

TABLE. . . . : KBDTSH
DATABASE . . : KBDD001
DESCRIPTION. : SCHEDULE TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTSH	ISYSIDY	CHAR	4	APPLICATION ID
	IOPUCTY	CHAR	3	COUNTRY CODE
	SKBH_SCHKEY	CHAR	8	SCHEDULE ID
	SKBH_SCHTXT	CHAR	50	SCHEDULE DESCRIPTION
	SKBH_SCHSTAT	CHAR	1	SCHEDULE STATE
	SKBH_MSGID	CHAR	10	MESSAGE TO BE ISSUED TO USER
	SKBH_MSGSUB1	CHAR	25	MESSAGE VARIABLE 1
	SKBH_MSGSUB2	CHAR	25	MESSAGE VARIABLE 1
	SKBH_SCHCOMM	CHAR	50	COMMENT ABOUT CLOSE
	SKBH_SCHDATA	VARCHAR	480	SCHEDULE DATA
	IUSRUUL	CHAR	8	USERID FOR LAST UPDATE
	DSYSRPT	TIMESTMP	10	TIME FOR LAST UPDATE

TABLE. . . . : KBDTSUN
DATABASE . . : KBDD001
DESCRIPTION. : SUNDRY TEXT TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTSUN	ISYSIDY	CHAR	4	APPLICATION ID
	IOPUCTY	CHAR	3	COUNTRY CODE
	CUSRLAN	CHAR	2	LANGUAGE CODE
	SKBH_SUNITEM	CHAR	8	SUNDRY ITEM NAME
	SKBH_SUNVALUE	CHAR	16	SUNDRY ITEM VALUE
	SKBH_SUNVAL_I	INTEGER	4	SUNDRY ITEM VALUE INTEGER
	SKBH_SUNVAL_D	DECIMAL	15	SUNDRY ITEM VALUE DECIMAL
	SKBH_SUNTEXT	VARCHAR	80	SUNDRY ITEM TEXT
	IUSRUUL	CHAR	8	USERID FOR LAST UPDATE
	DSYSRPT	TIMESTMP	10	TIME FOR LAST UPDATE

TABLE. . . . : KBDTSV
DATABASE . . : KBDD001
DESCRIPTION. : IMB SCOPE VALUE TABLE

TABLE	COLUMN	TYPE	LENGTH	DESCRIPTION
KBDTSV	IOPUCTY	CHAR	3	COUNTRY CODE
	ISCOIDY	CHAR	16	SCOPE KEY
	ISCOVAL	CHAR	24	SCOPE VALUE
	NSCOTXT	CHAR	40	DESCRIPTIVE TEXT
	IUSRUUL	CHAR	8	USERID OF LATEST UPDATE
	DSYSRPT	TIMESTMP	10	TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

```
TABLE. . . . : KBDTSX
DATABASE . . : KBDD001
DESCRIPTION. : IMB MAILROOM SOURCE EXIT

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTSX    SKBA_TYPE_SRC       CHAR           8  SOURCE SCENARIO TYPE
          SKBA_SENDER_ID1     CHAR          64  SENDER IDENTITY PART 1
          SKBA_SENDER_ID2     CHAR          64  SENDER IDENTITY PART 2
          SKBA_UNPACK_EXIT    CHAR           8  UNPACK EXIT
          SKBA_UNPACK_PARM    CHAR          40  UNPACK PARAMETER
          SKBA_SOURCE_EXIT    CHAR           8  SOURCE EXIT
          SKBA_SOURCE_PARM    CHAR          40  SOURCE PARAMETER
          IOPUCTY             CHAR           3  COUNTRY CODE
          SKBA_TPID_TO        CHAR          35  TO TRADING PARTNER (EXTERNAL)
          SKBA_TPID_FROM      CHAR          35  FROM TRADING PARTNER (EXTERNAL)
          SKBA_LAYOUT         CHAR          16  FORMAT OF THE DOCUMENT
          SKBA_REF_DATA       CHAR          40  REFERENCE DATA
          IUSRUUL             CHAR           8  USERID
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF INSERT TIME

TABLE. . . . : KBDTTA
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM TRANSPORT APPLDATA UNIQUE TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTTA    IOPUCTY             CHAR           3  TRADING PARTNER COUNTRY
          SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_DOCSEQNO       INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          SKBA_TYPE_SRC       CHAR           8  SOURCE SCENARIO TYPE
          SKBA_REF_DATA       CHAR          40  APPLICATION REFERENCE DATA

TABLE. . . . : KBDTTC
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM TRANSPORT TABLE CONTROL

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTTC    IOPUCTY             CHAR           3  TRADING PARTNER COUNTRY
          SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_DOCSEQNO       INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          ICUSPRM_TO          CHAR           9  TO TRADING PARTNER NUMBER
          ICUSPRM_FROM        CHAR           9  FROM TRADING PARTNER NUMBER
          SKBA_TPID_TO        CHAR          35  TO TRADING PARTNER (EXTERNAL)
          SKBA_TPID_FROM      CHAR          35  FROM TRADING PARTNER (EXTERNAL)
          SKBA_TYPE_SUB       CHAR           1  SUBSCRIPTION TYPE
          SKBA_LAYOUT         CHAR          16  DOCUMENT LAYOUT
          SKBA_REF_DATA       CHAR          40  APPLICATION REFERENCE DATA
          SKBA_TOTLIN         INTEGER        4  TOTAL NUMBER OF LINES IN DOCUMENT
          SKBA_MAXLNG         INTEGER        4  MAX LENGTH OF DATA RECORD
          SKBA_DOCSEQ_NEW     INTEGER        4  NEW DOCUMENT SEQUENCE NUMBER
          SKBA_DOCSEQ_OLD     INTEGER        4  OLD DOCUMENT SEQUENCE NUMBER
          SKBA_DOCVERSION     CHAR          30  DOCUMENT VERSION
          DSYSRPT             TIMESTMP      10  DATE, WHEN ENVELOPE WAS INS/UPD

TABLE. . . . : KBDTTG
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM TOTAL TRANSPORT TABLE CONTROL

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTTG    IOPUCTY             CHAR           3  TRADING PARTNER COUNTRY
          SKBA_ENVKEY         CHAR          12  MR ENVELOPE KEY
          SKBA_TOTDOC         INTEGER        4  TOTAL NUMBER OF DOCUMENTS IN ENV.
          SKBA_TOTIMG         INTEGER        4  TOTAL LMAGES OF DOCUMENT
          SKBA_TOTLOG         INTEGER        4  TOTAL LOGICAL DOCUMENTS
          SKBA_TYPE_SRC       CHAR           8  SOURCE SCENARIO TYPE
          SKBA_ORIGINATOR     CHAR          80  -ASSIGNED BY SOURCE API
          DSYSRPT             TIMESTMP      10  DATE, WHEN ENVELOPE WAS INSERTED
```

TABLE. . . . : KBDTTI

```
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM TRANSPORT TABLE INDEX

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTTI    IOPUCTY            CHAR           3  TRADING PARTNER COUNTRY
          SKBA_ENVKEY        CHAR          12  MR ENVELOPE KEY
          SKBA_DOCSEQNO      INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          SKBA_STATUS        CHAR          10  PROCESSING STATUS OF THE DOCUMENT
          SKBA_PROCTIME      TIMESTMP      10  PROCESS TIME FOR THE DOCUMENT
          SKBH_SCHKEY        CHAR           8  SCHEDULE ID
          SKBA_LAYOUT        CHAR          16  FORMAT OF THE DOCUMENT
          ISYSIDY            CHAR           4  APPLICATION PROJECT ID
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          SKBA_TYPE_DST      CHAR           8  DESTINATION SCENARIO TYPE
          SKBA_ENV_DST       CHAR          12  MR ENVELOPE KEY - DESTINATION
          SKBA_REQ_KEY       CHAR          40  MAILROOM REF.KEY (TO REQUEST TABLE)
```

TABLE. . . . : KBDTTR

```
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM TRANSPORT TABLE DOCUMENT DATA

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTTR    IOPUCTY            CHAR           3  TRADING PARTNER COUNTRY
          SKBA_ENVKEY        CHAR          12  MR ENVELOPE KEY
          SKBA_DOCSEQNO      INTEGER        4  DOCUMENT SEQUENCE NUMBER IN ENVELOPE
          SKBA_LINNO         INTEGER        4  LINE NUMBER IN DOCUMENT
          SKBA_DATAREC       VARCHAR     2000  DATA RECORD
```

TABLE. . . . : KBDTUA

```
DATABASE . . : KBDD001
DESCRIPTION. :  USER ACCESS TO SCOPE

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTUA    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IUSRIDY            CHAR           8  IMB USERID
          IKAAPRF            CHAR           8  LOCAL OPTION SET ID
          ISCOIDY            CHAR          16  SCOPE KEY
          ISCOVAL            CHAR          24  RESTRICTION FILTER
          CSCOPRI            CHAR           4  PRIVILEGE
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

TABLE. . . . : KBDTUP

```
DATABASE . . : KBDD001
DESCRIPTION. :  USER ACCESS TO LOCAL AGREEMENT SET

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTUP    IOPUCTY            CHAR           3  COUNTRY CODE
          ICUSPRM            CHAR           9  TRADING PARTNER NUMBER
          IUSRIDY            CHAR           8  IMB USERID
          IKAAPRF            CHAR           8  LOCAL OPTION SET ID
          IUSRUUL            CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT            TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

```
TABLE. . . . : KBDTUS
DATABASE . . : KBDD001
DESCRIPTION. : IMB USER TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTUS    IOPUCTY             CHAR           3  COUNTRY CODE
          ICUSPRM             CHAR           9  TRADING PARTNER NUMBER
          IUSRIDY             CHAR           8  IMB USERID
          IPRSIDY             INTEGER        4  PERSON ID (F.KEY TO CS-DB)
          IKAAPRF             CHAR           8  LOCAL OPTION SET ID (F.KEY TO KBDTPR)
          CUSRSEC             CHAR           1  LOCAL USER ADMINISTRATOR BIT
          CUSRLAN             CHAR           2  PREFERRED LANGUAGE
          CUSRDLM             CHAR           1  DELIMITER
          CUSRCSR             CHAR           1  CURSOR POSITION
          CUSRPOC             CHAR           1  PANEL-ID DISPLAY INDICATOR
          FUSRFID             CHAR           1  F-KEY DISPLAY CODE (NOT USED YET)
          IUSRPRM             CHAR           7  TEMP TRADING PARTNER NO. (NOT USED YET)
          CUSRIAS             CHAR           3  CLASSIFICATION CODE  (NOT USED YET)
          CUSRCTG             CHAR           1  USER CATEGORY (AGENT/DEALER.) (NOT USED YET)
          CUSRLVL             CHAR           3  LEVEL (NOT USED YET)
          IUSRHLN             CHAR           8  HOME NODE ID (NOT USED YET)
          IUSRHLT             CHAR           8  HOME TERMINAL ID (NOT USED YET)
          IUSRHLP             CHAR           8  HOME PRINTER ID (NOT USED YET)
          IUSRIDY_DIAL        CHAR           9  DIAL IBM USERID
          IUSRIDY_SAC         CHAR           9  SAC USERID
          IUSRIDY_ENG         CHAR           9  ENGINE USERID
          IUSRIDY_IMS         CHAR           9  IMS USERID
          NCUSDEPT            CHAR          30  DEPARTMENT
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTWC
DATABASE . . : KBDD001
DESCRIPTION. : WORKING CRITERIA TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTWC    IOPUCTY             CHAR           3  COUNTRY CODE
          SKBH_WCKEY          CHAR          16  KEY FOR INFORMATION
          SKBH_WCDATA         CHAR          32  INFORMATION TO BE HELD
          SKBH_WCTEXT         CHAR          78  SHORT DESCRIPTION
          IUSRUUL             CHAR           8  USERID FOR LAST UPDATE
          DSYSRPT             TIMESTMP      10  TIME FOR LAST UPDATE

TABLE. . . . : KBDTXM
DATABASE . . : KBDD001
DESCRIPTION. : MAILROOM XML DOCUMENT DEFS

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTXM    SKBA_XML_ROOT       CHAR          64  NAME OF XML ROOT ELEMENT
          SKBA_XML_DESCRIPT   CHAR          64  DESCRIPTION OF XML DOCUMENT TYPE
          SKBA_XML_IOPUCTY    CHAR          64  XML PATH TO COUNTRY CODE
          SKBA_XML_TPID_TO    CHAR          64  XML PATH TO TRADING PARTNER, TO
          SKBA_XML_TPID_FROM  CHAR          64  XML PATH TO TRADING PARTNER, FROM
          SKBA_XML_LAYOUT     CHAR          64  XML PATH TO LAYOUT
          SKBA_XML_REF_DATA   CHAR          64  XML PATH TO REFERENCE DATA
          IOPUCTY             CHAR           3  FIXED VALUE COUNTRY CODE
          SKBA_TPID_TO        CHAR          35  FIXED VALUE TRADING PARTNER, TO
          SKBA_TPID_FROM      CHAR          35  FIXED VALUE TRADING PARTNER, FROM
          SKBA_LAYOUT         CHAR          16  FIXED VALUE LAYOUT
          SKBA_REF_DATA       CHAR          40  FIXED VALUE REFERENCE DATA
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)

TABLE. . . . : KBDTXT
DATABASE . . : KBDD001
DESCRIPTION. : IMB EXTERNAL TRADING PARTNER TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBDTXT    IOPUCTY             CHAR           3  COUNTRY CODE
          ICUSPRM             CHAR           9  TRADING PARTNER NUMBER
          ICUSEXT             CHAR          35  EXTERNAL TRADING PARTNER ID
          ICUSDES             CHAR          35  EXTERNAL TRADING PARTNER DESCRIPTION
          IUSRUUL             CHAR           8  USERID OF LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP OF LATEST UPDATE (DB2 FORMAT)
```

```
TABLE. . . . : KBBTAPL
DATABASE . . : KBBD001
DESCRIPTION. : BEC APPLICATION TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBBTAPL   APPLCODE            CHAR          16  APPLICATION CODE (K)
          LOCATION            CHAR           8  LOCATION OF THE PGM (CTY CODE) (K)
          IAPLIDY             CHAR           4  APPLICATION ID, PROJECT
          DESTTYPE            CHAR          12  TYPE OF DESTINATION (FK)
          SYSID               CHAR          20  CONNECTION NAME / LU / IP ADR
          TRAN                CHAR          20  TRANSACTION/TP PGM TO ATTACH
          TRANPARM            CHAR          20  PARM TO TRX
          SERVMOD             CHAR           8  SERVICE MODULE IN BACK-END
          EXTAPPL             CHAR          16  BEC TO BEC REMOTE APPLCODE
          EXTLOC              CHAR           8  BEC TO BEC REMOTE LOCATION
          UIDSRCE             CHAR           1  TYPE OF USERID IN PROT
          UIDAPPL             CHAR           8  HARDCODE USERID
          CODEPAGE            CHAR           8  CODEPAGE CONV
          MEMALLOC            CHAR           1  MEMORY ALLOCATION MODE
          SCHECTRY            CHAR           3  SCHEDULE COUNTRYCODE ((FK))
          SCHEDUL             CHAR           8  SCHEDULE KEY ((FK))
          SPLITMOD            CHAR           1  SPLITMODE (2 TRX)
          ALTLOCAT            CHAR           8  ALTERNATIVE LOCATION - REROUTING
          IUSRUUL             CHAR           8  USERID LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP LATEST UPDATE

TABLE. . . . : KBBTDTY
DATABASE . . : KBBD001
DESCRIPTION. : BEC DESTINATION TYPE TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBBTDTY   DESTTYPE            CHAR          12  TYPE OF DESTINATION (K)
          DESTDESC            CHAR          40  DESCRIPTION OF THIS ENTRY
          PROTMOD             CHAR           8  NAME OF PROTOCOL MODULE
          COMMMOD             CHAR           8  NAME OF COMMUNICATION MODULE
          COMMQMOD            CHAR           8  NAME OF QUERY MODULE
          COMMOPTS            CHAR           8  COMMUNICATION OPTION KEY
          COMMPRM1            CHAR           8  COMMUNICATION PARAMETER 1
          COMMPRM2            CHAR           8  COMMUNICATION PARAMETER 2
          COMMPRM3            CHAR          40  COMMUNICATION PARAMETER 3
          COMMPRM4            CHAR          40  COMMUNICATION PARAMETER 4
          PROTPRM1            CHAR           8  PROTOCOL PARAMETER 1
          PROTPRM2            CHAR          40  PROTOCOL PARAMETER 2
          IUSRUUL             CHAR           8  USERID LATEST UPDATE
          DSYSRPT             TIMESTMP      10  TIMESTAMP LATEST UPDATE

TABLE. . . . : KBBTERR
DATABASE . . : KBBD001
DESCRIPTION. : BEC MESSAGE TABLE

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KBBTERR   MSGID               CHAR           6  MESSAGE ID
          MSGSEV              CHAR           1  SEVERITY
          RETURNCD            SMALLINT       2  RETURNCODE
          REASONCD            INTEGER        4  REASON CODE
          MSGTEXT             CHAR          55  MESSAGETEXT

TABLE. . . . : KAFT01
DATABASE . . : KAFD001
DESCRIPTION. : In this table the trading partners are maintained.

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KAFT01    COPUMCO             CHAR           3  Country code.
          ICUSIDY             INTEGER        4  Trading partner identifier.
          ICUSCID             CHAR           9  trading partner number.
          NCUSNME             CHAR          60  trading partner legal name.
          CCUSIAC             CHAR           1  Indication for activity.
          IUSRUUL             CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update of record.
```

```
TABLE. . . . : KAFT06
DATABASE . . : KAFD001
DESCRIPTION. : In this table the person information is maintained.

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  -------------------------------------------------------------
KAFT06    COPUMCO             CHAR           3  Coyntry code.
          IPRSIDY             INTEGER        4  Person identifier.
          NPRSNOP             CHAR          30  The persons surname.
          NPRSFOP             CHAR          30  The persons firstname.
          NPRSGEN             CHAR           5  Title (Mr., Miss., etc.)
          NPRSJBT             CHAR          30  Job title.
          NPRSTLE             CHAR          12  Professional title.
          IPRSMLP             CHAR           6  Mail point.
          ITLCEXT             CHAR           8  Telephone extension number.
          ITLCALT             CHAR           8  Alternative telephone extension number.
          CCUSIAC             CHAR           1  Indicator for activity.
          IUSRUUL             CHAR           8  record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update of record.

TABLE. . . . : KAFT09
DATABASE . . : KAFD001
DESCRIPTION. : In this table the information about the applications which has insert update-access to CS db
               is maintained.

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  -------------------------------------------------------------
KAFT09    COPUMCO             CHAR           3  Country code.
          IAPLIDY             CHAR           3  Application identifier.
          TAPLIDY             CHAR          20  Short description of application.
          IUSRUUL             CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update of record.

TABLE. . . . : KAFT13
DATABASE . . : KAFD001
DESCRIPTION. : In this table the person-use description is maintained.

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  -------------------------------------------------------------
KAFT13    COPUMCO             CHAR           3  Country code.
          CPRSUS1             CHAR           6  Code for use of person.
          TPRSUS1             CHAR          40  Description of code for use of person.
          IUSRUUL             CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update of record.

TABLE. . . . : KAFT20
DATABASE . . : KAFD001
DESCRIPTION. : In this table the different application which is using keys on Tra. P  in CS db, can make a
               reservation.

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  -------------------------------------------------------------
KAFT20    COPUMCO             CHAR           3  Country code
          ICUSIDY             INTEGER        4  Trading partner identifier.
          IAPLIDY             CHAR           3  Application identifier.
          ICUSCNT             INTEGER        4  Reservation counter.
          IUSRUUL             CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update of record.

TABLE. . . . : KAFT25
DATABASE . . : KAFD001
DESCRIPTION. : In this table the different applications which is using keys on persons in CS db, can reserve

TABLE     COLUMN              TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  -------------------------------------------------------------
KAFT25    COPUMCO             CHAR           3  Country code.
          IPRSIDY             INTEGER        4  Person identifier.
          IAPLIDY             CHAR           3  Application identifier.
          IPRSCNT             INTEGER        4  Reservation counter.
          CPRSUS1             CHAR           6  Code for use of person.
          IUSRUUL             CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT             TIMESTMP      10  Time of latest update.
```

```
TABLE. . . . : KAFT35
DATABASE . . : KAFD001
DESCRIPTION. : In this table the country information is maintained.

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KAFT35    COPUMCO            CHAR           3  Country code.
          IOPUISC            CHAR           2  Identifier of a country as defined by the ISO.
          NOPUCNL            CHAR          25  Country name in national language.
          NOPUCNM            CHAR          25  Country name in English.
          COPUCAN            CHAR           5  Abbreviated country name.
          CCINCUA            CHAR           3  Currency.
          IUSRUUL            CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT            TIMESTMP      10  Time of latest update of record.

TABLE. . . . : KAFT39
DATABASE . . : KAFD001
DESCRIPTION. : In this table the relation between the trading partner and the person  is maintained.

TABLE     COLUMN             TYPE      LENGTH  DESCRIPTION
--------  ------------------  --------  ------  ------------------------------------------------------------
KAFT39    COPUMCO            CHAR           3  Country code.
          IPRSIDY            INTEGER        4  Person identifier..
          ICUSIDY            INTEGER        4  Trading partner identifier.
          IUSRUUL            CHAR           8  Record latest updated by user-id or program-id.
          DSYSRPT            TIMESTMP      10  Time of latest update of record.
```

Appendix B. Recommended naming standards for CSP objects

Table 17. CSP naming standard

Object	Name	Maximum length
CSP Application	sssaaAP	7
CSP Browser-module	sssnnAP	7
CSP Update-module	sssyzAP	7
(DB2 Table)	(sssPROD.sssTyz)	
Map Group	sssaaM	6
Map	sssaaMxx	8
Help Map	sssaaHxx	8
Process	sssaaP_x..x	18
Statement Group	sssaaS_x..x	18
CSP Table	sssaaTx	7
Working Storage	sssaaWx	18
Redefined Working Storage	sssaaWxy	18
CSP file - DL/I record	sssaaFxx	8
CSP SQL record	sssaaRxx	8
CSP Global Item	xxxxxxx (data dict. items) sssaaIx..x (other)	32
CSP Local Item	sssaaIx..x or xxxxxx..x	32

Examples

CSP-APPLICATION

```
sss         =>  SYSTEM ID       =>  KBA
   aa       =>  APPLICATIONS ID =>  KBAYY
     AP     =>  CSP OBJECT TYPE =>  KBAYYAP
```

CSP-BROWSER MODULE

```
sss         =>  SYSTEM ID       =>  KBA
   nn       =>  APPLICATIONS ID =>  KBA01
     AP     =>  CSP OBJECT TYPE =>  KBA01AP
```

CSP-UPDATE MODULE (DB2TABLE: D123PROD.KBDTPR)

```
sss         =>  SYSTEM ID       =>  KBD
   yz       =>  TABEL suffix    =>  KBDPR
     AP     =>  CSP OBJECT TYPE =>  KBDPRAP
```

CSP-PROCESS

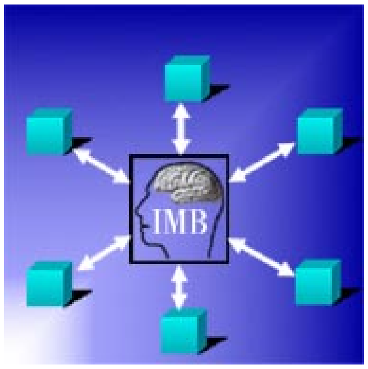
```
sss         =>  SYSTEM ID       =>  KBA
   aa       =>  APPLICATIONS ID =>  KBAYY
     P      =>  CSP OBJECT TYPE =>  KBAYYP
      _x..x =>  SUFFIX          =>  KBAYYP_MAIN
```

CSP-WORKING STORAGE

```
sss         =>  SYSTEM ID       =>  KBA
   aa       =>  APPLICATIONS ID =>  KBAYY
     W      =>  CSP OBJECT TYPE =>  KBAYYW
      x     =>  SUFFIX          =>  KBAYYWA
```

Index

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.



SC27-1583-00