CA Aion® Business Rules Expert

Mainframe User Guide

r11

CA Product References

This document references the following CA products:

- CA Aion® Business Rules Expert (CA Aion BRE)
- CA Aion® Multi-Tasking Aion Execution System (MAES)

Contact CA

Contact Technical Support

For your convenience, CA provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA product documentation, complete our short customer survey, which is also available on the CA Support website, found at http://ca.com/docs.

Contents

Chapter 1: Introduction

CA Aion Business Rules Expert provides an integrated development environment that allows organizations to build intelligent components and applications that can be flexibly deployed throughout the enterprise. Combining business rules technology and object-oriented programming, CA Aion Business Rules Expert (CA Aion BRE) allows companies to build maintainable applications that capture the knowledge and expertise of their organizations.

Aion

Aion is used when referring to the technology of Aion BRE. Aion BRE applications can be deployed as distributed components across a range of enterprise architectures and platforms.

This section contains the following topics:

CA Aion BRE Features

This guide describes the features of Aion BRE for IBM z/OS platforms, the Aion BRE Multi-tasking Aion Execution System (MAES) that allows multiple users to access a single Aion application in a shared address space, and the BMP/Batch Aion Execution System (BAES) that can be used by Aion applications to access IMS DL/1 databases.

Using CA Aion BRE on a Windows workstation and CA Aion BRE on the mainframe, you can create, test, debug, deploy, and maintain your knowledge based Aion applications. The following are significant features of CA Aion BRE:

Integrated Development Environment (IDE)

Development of Aion applications should take place on the Windows platform using the CA Aion BRE Integrated Development Environment (Aion IDE). The Windows-based IDE provides Remote Development facilities for other platforms.

Specific object editors

Use the editors provided by the Windows-based IDE to create and maintain the objects used by your applications.

Direct access

Direct access is supported for QSAM, DB2, and VSAM databases. Direct access means the data is accessed in the same region in which the Aion application is executing (that is, MAES, batch, or TSO region).

Indirect access

Indirect access is supported by the IMS and CICS data facilities. DL/I data is supported for both CICS and IMS, and VSAM and DB2 are supported for CICS.

XML parsing

Access XML content using either the Simple API for XML (SAX) or the Document Object Model (DOM) parsing techniques.

Batch application execution

Standalone Aion applications can be executed in batch or TSO environments. This is the simplest way to use Aion BRE capabilities in a mainframe environment

MAES (Multi-Tasking Aion Execution System)

MAES allows multiple users to concurrently access Aion applications in a shared address space through CICS, IMS message regions, or TCP/IP clients.

BAES (Batch Aion Execution System)

BAES can be used by your Aion applications to access IMS and DL/I databases.

About This Guide

The material in this guide assumes that the appropriate CA Aion BRE components have been installed in your environment. It also assumes that you have purchased CA Aion BRE on a Windows workstation and have access to the *CA Aion BRE Mainframe Installation Guide* and the *CA Aion BRE Product Guide*.

Detailed instructions for installing CA Aion BRE for mainframe platforms are provided in the *CA Aion BRE Mainframe Installation Guide*. For helpful information about starting and using CA Aion BRE, see the *CA Aion BRE Product Guide*.

This *CA Aion BRE Mainframe User Guide* contains information about Aion services available on the mainframe. It provides instructions you can use to examine and test your Aion applications. It also contains detailed information about how to create Aion components, and the use of those components by IMS, CICS, and PC clients.

Find the Information You Need

The information contained in this guide is presented in the order in which it is typically needed. The outline that follows shows the primary purposes for using CA Aion BRE and identifies the chapters that furnish information related to those tasks:

For Developing Applications

Chapter 2: Aion Applications Database Access (see page 13)

For Building Applications

Chapter 3: Build and Manage Aion Applications (see page 71)

Chapter 4: Build and Manage Aion Components (see page 107)

For Running Applications

Chapter 5: MAES (the Multitasking Aion Execution System) (see page 171)

Chapter 6: BAES (the BMP/Batch Aion Execution System) (see page 245)

For Reference

Appendix A: Accounting Record Format (see page 255)

Appendix B: Messages & Codes (see page 259)

Microsoft Windows

Unless otherwise indicated, the term Windows refers to any Microsoft Windows operating system supported by CA Aion BRE, including Windows XP, Windows Vista, Windows Server 2003, and Windows Server 2008. For more information, see CA Support Online for current information regarding supported windows environments.

Chapter 2: Aion Applications Database Access

This chapter provides information about how to use the various data access methods supported by CA Aion BRE for the mainframe.

This section contains the following topics:

Select a Data Access Method

You can write logic to access mainframe data using the methods defined in the IOLIB, DLILIB, or DATALIB libraries. The method you use depends on the type of data you are accessing. For sequential data access, use the methods provided in the IOLIB and DLILIB libraries. For SQL data access, use the methods provided in DATALIB. DATALIB is discussed in the *CA Aion BRE Product Guide.*

Direct and Indirect Access

CA Aion BRE supports direct file access for QSAM, DB2, and VSAM databases. Direct access means the data is accessed in the same region in which the Aion application is executing (that is, MAES, batch, or TSO region).

Indirect access is supported by the IMS and CICS data facilities. DL/I data is supported for both CICS and IMS. VSAM and DB2 are supported for CICS.

Using indirect file access through the CICS or IMS region, you can provide the following:

Transaction control

Commits and rollbacks (for example) can be handled automatically.

Security

Only authorized users can access and update the data.

The following table identifies the access methods supported by CA Aion BRE for each type of data:

Type of Data	Direct Access Support (in MAES)	Indirect Access Support	
		CICS	IMS
QSAM	X		
VSAM	X	X	
DL/I			X
DB2	X	X	

QSAM File Access

QSAM is a file management system used to manage and access files stored on a mainframe. You can use QSAM to access sequential data sets and partitioned data set (PDS, PDS/E) members under Z/OS. QSAM records are ordered in the sequence in which they were written to the file.

Using Aion BRE, you can retrieve, modify, and add QSAM records. You can read and write both text and binary data. You cannot delete QSAM files using CA Aion BRE. QSAM files can only be accessed from TSO, batch, or locally within the MAES region. There is no support for indirect QSAM data access from CICS or IMS.

Examples of QSAM File Access can be found in <AIONHLQ>.SAMPLES.JCL.QSAM & refer to <aionhlq>.SAMPLES.REDME(QSAM).

Important! From within MAES, you should not call external COBOL programs that contain QSAM file access. COBOL is not reentrant and QSAM file access is not supported by Z/OS under these circumstances.

QSAM File Access Methods

To access a QSAM file from CA Aion BRE, compose your logic using methods contained in the IOLIB File class. You can use these methods to:

- Open or close a QSAM file
- Position to a specific record in a file
- Retrieve (read), add (write), modify (update), and delete records
- Check the location of the QSAM file pointer and reposition as needed
- Implement customized error processing

The following lists QSAM-related methods in the IOLIB and the task each method performs:

Open

Open a file to read, write, or append

Close

Close a file

EOF

Check for end-of-file

This check returns true if the file pointer is positioned at the end of the file; otherwise, the check returns false

Read

Read the next sequential record in the file into a string buffer

ReadBinary

Read the next sequential record in the file into a binary buffer

Write

Write the string buffer as the next record of the file

WriteBinary

Write the binary buffer as the next record of the file

GetPosition

Get the position of the record pointer inside a file relative to the beginning of the file

This position can be used later as a displacement argument in the Reposition method

Reposition

Reposition the record pointer inside a file, for example, when changing from a read to a write operation

WhenError

This method is automatically invoked when unexpected I/O errors are encountered on Read, Write, Close, or EOF

If the file I/O operation results in a nonzero return code, it is passed as the return code input argument. To do your own customized I/O error handling, subclass the appropriate File class and specialize this method appropriately.

Open a QSAM File: Open Method

Use the Open method to open a QSAM file.

The syntax for this method is:

```
Open(filename,OPEN_mode[+filetype])
```

where:

filename

Is the name of the file to open, and is expressed as a string. This can be the actual file name, or it can be an input variable.

mode

Is the open mode expressed as one of the following:

READ

Allows read-only retrieval of the record.

WRITE

Destroys the contents of the file and allows you to add new records.

APPEND

Allows you to insert new records into the file without destroying those that may already exist.

filetype

Is the type of file you are opening, and is expressed as one of the following:

FLAG_TEXT

Indicates that the file is a text file and contains no binary data.

FLAG_BINARY

Indicates that the file contains binary data. This is the default if *filetype* is omitted.

The following logic illustrates an example of the Open method:

```
/* This logic opens a QSAM file */
var pEmp is pointer to currentclass
pEmp = create
// Build a record to add to end of file
pEmp.Citizen        = true
pEmp.DateHired      = currentdate
pEmp.DateLastReview = currentdate
pEmp.NameFirst      = "Leo"
pEmp.NameLast       = "Lyons"
var rc is integer
rc = pFile.Open("DD:EMPFile",OPEN_APPEND+FILE_TEXT)
If (rc<>IO_OK) then
    return
End
pEmp.WriteEmployee
rc = pFile.Close
If (rc<>IO_OK) then
    return
End
```

Close a QSAM File: Close Method

Use the Close method to close a QSAM file.

The syntax for this method is:

```
Close
```

The following logic illustrates an example of the use of the Close method:

```
/* This logic closes a QSAM file */
var rc           is integer
var pEmp         is pointer to Currentclass
rc = pFile.Open(pFile.FileName,Open_write+FileType)
If (rc<>IO_OK) then
    return
End
for currentclass, pEmp
    pEmp.WriteEmployee
end
rc = pFile.Close
If (rc<>IO_OK) then
    return
End
```

Check for the End of a File: EOF Method

Use the EOF method to determine whether the file pointer is positioned at the end of a file. The method returns true if the current position is at the end of the file; otherwise, the method returns false.

Retrieve a Text Record: Read Method

Use the Read method to retrieve a *text* record from an open file. The record pointer starts at the first record in the file and increments sequentially forward to the end of the file (unless a Reposition statement is used to explicitly position the file pointer).

The syntax for this method is:

Read(*output_string*)

output_string

Is an output string with a maximum size of 64 KB.

If the record contains fields of data, use *SubString* and *decode* to break it into fields and attributes, that is, substring *source* from *start* for *length* characters.

The following logic illustrates an example of the use of the Read method:

```
/* This logic retrieves a QSAM text record */
var rec, fld is string
var rc integer
var position is integer = 0
rc = pFile.Read(rec)
If (pFile.EOF) then
    return(FALSE)
end
fld = format(SubString(rec,1,10),FMT_FLAG_TRIM)
NameFirst = fld
fld = format(SubString(rec,11,10),FMT_FLAG_TRIM)
NameLast  = fld
fld = SubString(rec,21,2)
if (NOT (decode(VacationDays,fld))) then
    DecodeError("Vacation Days='" & fld & "'")
end
fld = SubString(rec,23,1)
if (NOT (decode(Citizen,fld))) then
    DecodeError("Citizen='" & fld & "'")
end
return(TRUE)
```

Retrieve a Binary Record: ReadBinary Method

Use the ReadBinary method to retrieve a *binary* record from an open file. The record pointer starts at the first record in the file and increments sequentially forward to the end of the file (unless a Reposition statement is used to explicitly position the file pointer).

The syntax for this method is:

```
ReadBinary(buf,buflen)
```

buf

Is a binary output variable (maximum size 64 KB).

buflen

Is an integer input variable representing the maximum length of the record, and is optional on Z/OS. If omitted, only one record will be read.

You do not need to supply an OpenBinary method to initialize the binary buffer; the ReadBinary method does this for you.

The following logic illustrates an example of the use of the ReadBinary method:

```
/* This logic reads a QSAM binary record */
var rec  is binary
var rc      integer
var PackDec integer
if (OnMainframe) then
    rc = pFile.ReadBinary(REC)
else
    rc = pFile.ReadBinary(REC,RECLEN)
end
If (pFile.EOF) then
    return(FALSE)
end
If (rc<>IO_OK) then
    return(FALSE)
End
rc = ReadString(rec,NameFirst,0)
rc = ReadString(rec,NameLast,0)
rc = ReadInteger(rec,VacationDays,bin_fmt_zoned_dec,4)
//Total deductions is a packed decimal with two decimal places
//A placeholder is used until the decimal point is calculated
rc = ReadInteger(rec,PackDec,bin_fmt_packed_dec,5)
TotalDeductions = PackDec/100
```

Write a Text Record: Write Method

Use the Write method to write a *text* record to a QSAM file. You can specify whether CA Aion BRE should append the record to the end of the QSAM file, or update the original record.

The syntax for this method is:

```
Write(input_string,input_lf)
```

input_string

Is the input string.

input_lf

Specifies whether to write a line feed after writing the string.

If *input_string* is NULL and *input_lf* is true, only a carriage return/line feed is written to the file. If the record contains multiple fields, use the format method *Concat(&)* to build a formatted record.

The following logic illustrates an example of the use of the Write method:

```
/* This logic writes a text record to a QSAM file */
rec = format(NameFirst,NULL,NULL,10)
    &Format(NameLast,NULL,NULL,10)
    &Format(VacationDays,fmt_flag_leadingzeroes,null,-2)
        //width 2, right justify
    &Format(TotalDeductions,fmt_Real_fixed+fmt_flag_leadingzeroes
    ,null,-9)
var rc integer
rc = pFile.Write(rec)
If (rc<>IO_OK) then
    return(FALSE)
End
    return(TRUE)
```

Write a Binary Record: WriteBinary Method

Use the WriteBinary method to write a *binary* record to a QSAM file.

The syntax for this method is:

```
WriteBinary(buf)
```

buf

Is a binary input variable.

The following logic illustrates an example of the use of the WriteBinary method:

```
/* This logic writes a QSAM file record.*/
/*The COBOL definition of the record layout is:*/
/*
01  WS-QSAM-EMPL-RECORD.
    10 WS-EMPL-FIRST    PIC X(10).
    10 WS-EMPL-LAST     PIC X(10).
    10 WS-EMPL-VAC-DAY  PIC S9(04).
    10 WS-EMPL-S-BASE   PIC 9(12)      COMP-3.
    10 WS-EMPL-T-DED    PIC S9(06)V99  COMP-3.
*/
var rec  is binary
var rc   is integer
var s    is string
var PackDec integer
rec = OpenBinary(Bin_Open_Write,RECLEN)
// Pad first name to 10 characters
s = format(NameFirst,NULL,NULL,10)
rc = WriteString(rec,s,0)
// Pad last name to 10 characters
s = format(NameLast,NULL,NULL,10)
rc = WriteString(rec,s,0)
rc = WriteInteger(rec,VacationDays,bin_fmt_zoned_dec,4)
rc = WriteInteger(rec,trunc(SalaryOvertime*100),bin_fmt_short)
// Convert real to packed decimal before writing to file
PackDec = trunc(TotalDeductions * 100)
rc = WriteInteger(rec,PackDec,bin_fmt_packed_dec,5)
rc = pFile.WriteBinary(rec)
If (rc<>IO_OK) then
    return(FALSE)
End
    return(TRUE)
```

Find the QSAM File Pointer Position: GetPosition Method

Use the GetPosition method to locate the position of the file pointer inside a QSAM file.

The syntax for this method is:

`GetPosition(`*`out_position`*`)`

Out_position

Is an integer specifying the number of the record on which the pointer is positioned (as counted from the beginning of the file).

Note: This position can be used later for the displacement argument in a Reposition method.

Reposition the File Pointer: Reposition Method

Use the Reposition method to change the position of the file pointer inside a QSAM file.

The syntax for this method is:

`Reposition(`*`origin`*`,`*`displacement`*`)`

origin

Must be one of the following values:

POS_FIRST

Specifies the first position in the file.

POS_CURRENT

Specifies the current position in the file.

POS_LAST

Specifies the last position in the file.

displacement

Is an integer specifying the record to which the pointer is repositioned.

The following logic illustrates an example of the use of the Reposition method:

```
/* This logic changes the file pointer position from the current
record in the file to the third record in the file */
// select record number 3
rc = pFile.reposition(POS_CURRENT, 3)
If (rc<>IO_OK) then
    return
End
```

Processing Error: WhenError Method

This method is automatically invoked when I/O errors occur on Read, Write, Close, or EOF operations.

Each IOLIB method returns a code that indicates the status of a file I/O operation. If a nonzero code, it is passed as the return code input argument. You can test this value to perform error handling. To do your own customized I/O error handling, subclass the appropriate File class and specialize this method appropriately.

The most common return codes have constants in IOLIB (for example, Z/OS_FILE_NOT_FOUND, IO_ERROR, and so on). By default, IOLIB sends write-error messages to SYSOUT.

The syntax for this method is:

```
WhenError(input_string,input_rc)
```

input_string

The input string for the system message.

input_rc

The input integer for the return code.

The following logic illustrates an example of a customized error handling using the WhenError method:

```
var msg string
msg = "File I/O Error - " & sysmsg
if (rc <> NULL) then
    // format return code and meaning of RC
    var RCText is string
    If (OnMainframe) then
        if RC = MVS_DDN_NOT_FOUND    then RCText = ", File DD name
        not found"
        // Do not expect to get here for EOF.
        // App should check (pFile.EOF) after read and return if true
        // before testing for other errors.
        ElseIf RC = IO_Error            then RCText = "I/O Error"
        ElseIf RC = MVS_EOF            then RCText = "End of file"
        ElseIf RC = MVS_File_Already_Opened then RCText = "File
        already open"
        ElseIf RC = MVS_File_Conflict    then RCText = "File open
        mode conflict"
        ElseIf RC = MVS_File_Notfound     then RCText = "File not
        found"
        ElseIf RC = MVS_Invalid_File      then RCText = "File is
        invalid"
        ElseIf RC = MVS_Invalid_Mode     then RCText = "File open
        mode invalid"
        ElseIf RC = MVS_IO_Syserr      then RCText = "System error"
        end
    msg = msg & ", Return Code: " & format(RC)
    if (length(RCText) > 0) then
        msg = msg & " ("& RCText&")"
    end
end
LastError = msg & ", File Name: " & FileName
if (NOT DisableErrorMessages) then
    LogError(LastError,"WhenError",currentclass,AbortOnError)
End
```

VSAM Data Access

Virtual System Access Method (VSAM) is a file management system used to manage and access data on the mainframe.

To access VSAM data, you can use or customize the supplied methods in the IOLIB VSAM File class.

Examples of VSAM DATA access can be found in <aionhlq>.JCL.VSAM and refer to <aionhlq>.SAMPLES.README(VSAM).

Supported VSAM Access Modes

CA Aion BRE supports the following VSAM access modes:

Addressed

Access records based on their physical position in the data set

Keyed

Access records using a key field contained in the data record

Sequential

Sequentially access records in a forward or backward direction

Direct

Directly access records using addressed-direct or keyed-direct positioning

Indirect

Indirectly access records through CICS from a PC or from MAES

Supported Data Set Types

Use access modes to retrieve data from the following VSAM data set types:

- Entry Sequenced Data Set (ESDS)-Records in an ESDS are sequenced in the order in which they were written to the data set. You can add, retrieve, and modify records, but you cannot delete records from the data set.

  The default retrieval mode is addressed-sequential, starting with relative byte address (RBA) 0 (zero). The RBA for a record is the physical offset from the beginning of the data set (initial value is 0). Use the addressed-sequential mode to process records sequentially (either forward or backward). Use the RBA to directly access a record.

- Key Sequenced Data Set (KSDS)-Records in a KSDS are sequenced by a key within each record. The key is of fixed length and is in the same position within each record. The key is unique and cannot be changed once it is stored in the record. You can create alternate indexes to allow duplicate alternate keys.

  You can add, retrieve, modify, and delete records. The default retrieval mode is keyed-sequential, starting with the record having the lowest key value.

  KSDS supports addressed-sequential and addressed-direct access to records; both methods use RBA addressing. It also allows direct or sequential record processing using the key value, which is keyed-direct or keyed-sequential processing, respectively.

- Relative Record Data Set (RRDS)-An RRDS is divided into slots where records are stored. Only one record can be stored in a slot. Typically, there are empty slots in an RRDS. You can add, retrieve, modify, and delete records in an RRDS.

  Records are identified by a key, which is the relative record number (RRN). The RRN corresponds to the slot number, such that the first slot is RRN 1, the second slot is RRN 2, and so on. The RRN key is maintained by VSAM.

  Access records using keyed-sequential processing or keyed-direct processing, where RRN is the key. If you use keyed-direct processing, the RRN is the selection argument. The initial retrieval mode is keyed-sequential starting with the first nonempty slot.

  Note: You cannot use RBA processing with RRDS.

VSAM File Access Methods

Use the methods in the IOLIB and VSAMFile classes to perform the following tasks:

- Open a VSAM file
- Position to a specific record in a file
- Select a subset of records in a file
- Retrieve (read), add (write), modify (update), and delete records

Note: For VSAM Read and Write operations, the maximum record length is 1024. For longer records, use the ReadBinary and WriteBinary methods. Attempts to use the Read and Write operations with records longer than 1024 will result in a return code of -3.

The following table lists VSAM-related methods in the VSAMFile class and the task each method performs:

Open

Open a file.

SelectRecordByKey

Select a specific record in a text file, and specify the direction (forward or backward) of the next read/write (based on the Select_Option).

SelectRecordByKeyBinary

Select a specific record in a binary file, and specify the direction (forward or backward) of the next read/write (based on the Select_Option).

SelectRecordByRBA

Select a specific record using its relative byte address (RBA), and specify the direction (forward or backward) of the next read/write (based on the Select_Option).

SelectRecordByRRN

Select a specific record using its relative record number (RRN), and specify the direction (forward or backward) of the next read/write (based on the Select_Option).

UpdateRecord

Modify a text record pointed to by a SelectRecordBy method.

UpdateRecordBinary

Modify a binary record pointed to by a SelectRecordBy method.

DeleteRecord

Delete a record pointed to by a SelectRecordBy method.

For more information about the IOLIB VSAMFile class, see the CA Aion BRE Online Help.

Open a VSAM File: Open Method

Use the Open method to open a VSAM file using direct access. You may perform indirect VSAM file access through CICS from a PC or from MAES by specifying a CICS region LU name in this method instead of a file type.

The syntax for this method is:

`Open(`*`filename`*`,OPEN_`*`mode`*`,`*`filetype`*`)`

filename

The name of an Z/OS DD in the format dd:ddname, or a VSAM file name in the format, xxx.yyy.zzz.

Important! Indirect VSAM access of files in CICS must be opened using a DD name.

mode

Is the open mode expressed as one of the following:

READ

Allows read-only retrieval of the data set record.

WRITE

Destroys the content of a data set and allows you to add new records.

APPEND

Allows you to insert new records (without destroying existing data) into a data set that may already contain records.

Important! If you are updating the record, it must be opened using the READ or APPEND mode.

filetype

Use the following conventions to indicate the type of file with which you are working:

FILE_TEXT

When added to the OPEN_*mode* constant (for example, OPEN_WRITE+FILE_TEXT), this file type indicates that the file is a text file and contains no binary data.

FILE_BINARY

When added to the OPEN_ *mode* constant (for example, OPEN_WRITE+FILE_BINARY), this file type indicates that the file contains binary data. This is the default file type if you do not specify *filetype*.

servername

This is the *LUNAME* of the CICS region where the file is to be accessed.

The following logic illustrates an example of enabling indirect VSAM file access by specifying the CICS LU name *CICSJLI* as the server:

```
/* This logic indirectly accesses (opens) a file on CICSJLI */
var pEmployee is pointer to currentclass
// Set text file name
SetupFile( )
if NOT
(pFileHandle.Open(pFileHandle.FileName,Open_Append,"CICSJLI") =
IO_OK )
then
    MessageBox( "ADD   : operation failed. Unable to open Employee
    file", "" )
    return
end
```

Select Text Records by Key: SelectRecordByKey Method

Prior to a Read or Write operation, use the SelectRecordByKey method to select a text record having a specific key value.

The syntax for this method is:

```
SelectRecordByKey(key,operation,keylen)
```

key

Identifies the VSAM record you wish to select.

operation

Determines the position in the file and whether the subsequent Read or Write operation will be forward or backward. The value may be one of the following:

KEY_EQ

Specifies the position to the first record in the VSAM file containing *key*. Subsequent reads are in the forward direction.

KEY_EQ_BWD

Specifies the position to the first record in the VSAM file containing *key*. Subsequent reads are in the backward direction.

KEY_FIRST

Specifies the position to the very first record in the VSAM file (*key* and *keylen* are ignored). Subsequent reads are in the forward direction.

KEY_GE

Specifies the position to the first record in the VSAM file containing a key value greater than or equal to *key.*

KEY_LAST

Specifies the position to the very last record in the VSAM file (*key* and *keylen* are ignored). Subsequent reads are in the backward direction.

keylen

Indicates the length of the key.

The following logic illustrates an example that positions to the first text record of the file containing the key, 100010:

```
var rc is integer
rc = pFileHandle.Open(pFileHandle.FileName,Open_Read,"EMPDJRK")
If (rc<>IO_OK) then
    return
End
pFileHandle.SelectRecordByKey("100010", KEY_EQ )
var pEmp pointer to currentclass
Loop
    pEmp = currentclass.create
    If (NOT (pEmp.ReadEmployee("100018"))) then
        pEmp.Delete
        break
    end
End
```

Select Binary Records by Key: SelectRecordByKeyBinary Method

Prior to a Read or Write operation, use the SelectRecordByKeyBinary method to select a binary record having a specific key value.

The syntax for this method is:

```
SelectRecordByKeyBinary(key,operation,keylen)
```

key

Identifies the VSAM record you wish to select.

operation

Determines the position in the file and whether the subsequent Read or Write will be forward or backward. The value may be one of the following:

KEY_EQ

Specifies the position to the first record in the VSAM file containing *key*. Subsequent reads are in the forward direction.

KEY_EQ_BWD

Specifies the position to the first record in the VSAM file containing *key*. Subsequent reads are in the backward direction.

KEY_FIRST

Specifies the position to the very first record in the VSAM file (*key* is ignored). Subsequent reads are in the forward direction.

KEY_GE

Specifies the position to the first record in the VSAM file containing a key value greater than or equal to *key.*

KEY_LAST

Specifies the position to the very last record in the VSAM file (*key* is ignored). Subsequent reads are in the backward direction.

keylen

Indicates the length of the key.

Select Text Records by RBA: SelectRecordByRBA Method

Prior to a Read or Write operation, use the SelectRecordByRBA method to select a record having a specific relative byte address.

The syntax for this method is:

`SelectRecordByRBA(rba,operation)`

rba

Is a value identifying the relative byte address. If you specify zero (0), CA Aion BRE points to the first record in the VSAM file.

operation

Determines whether the next read or write will be forward or backward. The value may be one of the following:

RBA_EQ

Specifies the positions to the record in the VSAM file with the specified *rba.* Subsequent reads are in the forward direction.

RBA_EQ_BWD

Specifies the positions to the record in the VSAM file with the specified *rba*. Subsequent reads are in the backward direction.

Select Text Records by RRN: SelectRecordByRRN Method

Prior to a Read or Write operation, use the SelectRecordByRBA method to select a text record having a specific relative record number.

The syntax for this method is:

`SelectRecordByRRN(`*`rrn,operation`*`)`

rrn

Is the relative record number.

operation

Determines whether the next read or write will be forward or backward.

Formatting the Data Buffer

After selecting data, format the data buffer. You can use this example, located in the SetEmployee method of the VSAM.APP sample application, as a model for formatting a buffer.

This code fragment formats a buffer named EmployeeRecord:

```
Var Citizenship    string
var iDate, iPackedDecimal, integer
ReadString(EmployeeRecord, NameFirst,  0, 20)
ReadString(EmployeeRecord, NameLast,   0, 40)
ReadInteger(EmployeeRecord, VacationDays, bin_fmt_zoned_dec, 3 )
ReadInteger(EmployeeRecord,iPackedDecimal, bin_fmt_packed_dec, 5)
TotalDeductions = iPackedDecimal / 100
return(TRUE)
```

Update a VSAM Record: UpdateRecord Method

Use the UpdateRecord method to update a text record pointed to by SelectRecordBy…. A Read operation must precede this UpdateRecord.

The syntax for this method is:

```
UpdateRecord(input_string)
```

input_string

The input string.

The following logic illustrates an example of updating the record just selected by the SelectRecordByKey method:

```
var
pEmployee    pointer to currentclass,
sEmployeeID string(6) = “200000” // ID of the employee to be updated
Currentclass.SetupFile( )
if NOT (pFileHandle.Open(pFileHandle.FileName,
OPEN_UPDATE,”CICSJLI”)= IO_OK) then
    MessageBox( “UPDATE FAILED: Unable to open Employee file for
    UPDATE. “, “” )
    return
end
if pFileHandle.SelectRecordByKey( sEmployeeID , KEY_EQ ) = IO_OK then
    pEmployee = currentclass.create
    pEmployee.EmployeeID = sEmployeeID
    pEmployee.UpdateEmployeeRecord( )
else
    MessageBox( “UPDATE FAILED: Unable to select employee record for
    UPDATE. “, “”
)
end
    if NOT (pFileHandle.Close( ) = IO_OK) then
        MessageBox( “UPDATE FAILED: Unable to close Employee file.
        “, “” )
    end
return
```

CA Product References

Use the UpdateRecordBinary method to update a binary record selected by SelectRecordBy. A ReadBinary operation must precede this UpdateRecordBinary.

The syntax for this method is:

```
UpdateRecordBinary(bufname)
```

bufname

The name of a buffer.

The following logic illustrates an example of updating the binary record that has the buffer name, EmployeeRecord:

```
Var
EmployeeRecord binary
    if NOT (pFileHandle.ReadBinary(EmployeeRecord) = IO_OK) then
        return FALSE
    end
    // set EMPLOYEE instance
    SetEmployee(EmployeeRecord)
    // set record to be written to EMPLOYEE file
    Salary          = Salary * 1.08
    SalaryBonus     = 0
    SalaryOvertime  = 0
    TotalDeductions = 1300.00
    SeekBinary( EmployeeRecord, 0, FALSE )
    // reset EmployeeRecord buffer
    SetEmployeeRecord(EmployeeRecord)
    // Update record in EMPLOYEE file
    if pFileHandle.UpdateRecordBinary(EmployeeRecord) = IO_OK then
        MessageBox( “     :  Employee “ & EmployeeID & “ UPDATED
        successfully” , “”
)
        return TRUE
    else
        MessageBox( “ERROR:  Employee “ & EmployeeID & “ NOT UPDATED”
        , “” )
        return FALSE
    end
```

Delete a VSAM Record: DeleteRecord Method

Use the DeleteRecord method to delete a VSAM record pointed to by a preceding SelectRecordBy operation.

The syntax for this method is:

DeleteRecord

Note: Do *not* perform a Read between the SelectRecordBy and the DeleteRecord operations.

The following logic illustrates an example of using the DeleteRecord method to delete a VSAM employee record:

```
/* This logic deletes the VSAM record for employee ID 200000 */
var EmployeeID    string(6) = “200000”
SetupFile( )
// open EMPLOYEE file
if NOT
(pFileHandle.Open(pFileHandle.FileName,OPEN_UPDATE,”CICSJLI”) =
IO_OK )
then
    return
end
// delete EMPLOYEE record
if pFileHandle.SelectRecordByKey( EmployeeID, KEY_EQ ) = IO_OK then
    if pFileHandle.DeleteRecord( ) = IO_OK then
        MessageBox( “     :  Employee “ & EmployeeID & “ DELETED
        successfully”,””
)
    end
end
// close EMPLOYEE file
pFileHandle.Close( )
return
```

DL/I Data Access

DL/I (Database Language 1) is a database access method. You can access DL/I databases from Aion/IMS, Aion/CICS, or from the PC. CA Aion BRE provides methods in DLILIB to perform DL/I calls.

CA Aion BRE also supports the use of *native* DL/I statements. By calling the ExDliRequest method found in the DLILIB Task class, you utilize native DL/I statements in the CA Aion BRE environment. The use of this method depends on the correct environmental settings. Review these settings by using the higher-level methods in DL/I.

Define DL/I Data Structures to CA Aion BRE

DL/I databases are organized in a hierarchical structure. A database description (DBD) describes the physical structure of the database. DBDs are written in DL/I, and describe the size, name, position, data type, and other data segments in the hierarchy.

The following is an example of logic that defines a DBD:

```
DBD
    NAME=CPODBD1,ACCESS=(HDAM,VSAM),
    RMNAME=(DFSHDC40,1,300)
DATASET    DD1=CPODBD1,DEVICE=3390
SEGM    NAME=CPODBD1,BYTES=255,PARENT=0
FIELD    NAME=(IDNR,SEQ,U),BYTES=8,START=1,TYPE=C
DBDGEN
FINISH
END
```

Segment Search Arguments

Segment search arguments (SSAs) are used to position to a specific segment within a database.

The following lists the segment search argument methods:

AddSSA

Adds one SSA at index to the list of SSAs within a DL/I instance

DeleteSSA

Removes an SSA at index from the SSA list

DL/I Data Access Methods

Using the methods contained in the DLILIB DLI class, you can access DL/I data and perform the following operations:

- Open DL/I segments.
- Close DL/I segments.
- Retrieve selected segments.
- Insert or update segments into the database.
- Delete segments.
- Build a DL/I search argument.

The following lists the DL/I-related methods in DLILIB and their purpose:

Open

Connects to the database server, and presets the access mode to UPDATE or READ

Close

Kills the connection to the database server

Get

Reads one segment from the database

Put

Updates or inserts one segment within the database

Erase

Deletes the current segment from the database

EOF

Detects the segment end and returns status code GE or GB

SetConnectionInfo

Specifies the parameters for the APPC connection to the database server

Open DL/I Segments: Open Method

Use the Open method to connect to the database server and preset the access mode to UPDATE or READ.

The syntax for this method is:

Open(dbname, servernm, mode)

dbname

Is the IMS database name, using a maximum of eight characters.

servernm

Is the name of a connection entry, such as HOST1.

mode

Is either READ or UPDATE.

Close DL/I Segments: Close Method

To close DL/I segments, use the Close method.

The syntax of this method is:

`Close(`*`dbhandle`*`)`

dbhandle

The database handle (returned by the Open method).

Retrieve Selected Segments: Get Method

The Get method reads one segment from the database. It uses the SSAs previously set by the AddSSA segment search argument.

The syntax for this method is:

Get(dbhandle, buf, options)

dbhandle

Is the handle returned by the Open method.

buf

Is the data buffer (binary).

Important! The data buffer must be the same size as the segment in the database.

options

Can be the following:

PARENT

Specifies that the GET request will contain the parent command P, that is, GHNP. The P command is also used if an SSA contains a parent command code *P.

UPDATE

Specifies that the GET request is issued with HOLD.

Insert or Update Segments Within the Database: Put Method

Use the DLILIB Put method to update or insert a segment within the database.

The syntax for this method is:

Put(*dbhandle, buf, mode*)

dbhandle

Is the handle returned by the Open method.

buf

Is the data buffer (binary).

mode

Can be one of the following:

ADD

Inserts a segment.

UPDATE

Replaces a segment.

Delete Segments: Erase Method

Use the Erase method to delete the current segment from the database.

The syntax for this method is:

Erase(*dbhandle, buf*)

dbhandle

Is the database handle (returned by the Open method).

buf

Is the data buffer (binary).

Example: DL/I Methods

The following example illustrates the use of the Open, Close, Get, Put, and Erase methods:

```
var pDLI is pointer to DLI
var inbuf is binary(256)
var hDB is integer
var data is string
var count is integer
var pcbstatus is string
inbuf = OpenBinary(BIN_OPEN_WRITE,256)
pDLI=DLI.Create( )
hDB=pDLI->Open("CPODBD1 ","RAES","UPDATE")
if (hDB > 0) then
    pDLI->AddSSA("CPODBD1 ","(IDNR     =LAST2  F)",1)
//pDLI->AddSSA("CPODBD2 ","(CPODBD2  =00000002)",2)
    pDLI->Get(hDB,inbuf," ")
    pcbstatus=pDLI->GetPcbStat( )
    ReadString(inbuf,data,0)
//pDLI->Ebcdic2Ascii(data,Length(data))
//MessageBox(data,"Info",MBS_OK)
    pDLI->DeleteSSA(1)
    pcbstatus=pDLI->GetPcbStat( )
    data = "LAST2  FIRST2 INSERTED DATA"
//pDLI->Ascii2Ebcdic(data,Length(data))
    count=WriteString(inbuf,data,0)
    pDLI->AddSSA("CPODBD1 ","(IDNR     =LAST2  F)",1)
    pDLI->Put(hDB,inbuf,"UPDATE")
    pcbstatus=pDLI->GetPcbStat( )
    pDLI->Erase(hDB,inbuf)
    pDLI->Close(hDB)
end
pDLI->Delete( )
return 0
```

DL/I Error Processing Methods

DLILIB contains the following error processing methods:

GetLastStatusCode

Returns the last two-byte status code issued by IMS

GetPcbStat

Returns the IMS PCB status code.

EOF

IMS returns status codes GE or GB when it reaches the logical end of the file.

DB2 Data Access

To access data in a DB2 database, use the methods in DATALIB. This library is discussed in the *CA Aion BRE Product Guide.*

Note: In CA Aion BRE for the mainframe only, Aion/PC times cannot map to the DB2/PC data type, TIME. In CA Aion BRE, a TIME data type is implemented as a DB2 Date-Time. This means an Aion Time data type can be written to a DB2 timestamp, but not to a DB2 Time. You can read a DB2 Time into an Aion Time, but to update a DB2 Time variable, you must map the Time to an Aion String attribute and format the appropriate Time value (such as 05.15.45.0).

Depending on your DB2 ODBC installation, you may find it necessary to specify a DSNAOINI file in any JCL or TSO environments used to run Aion applications that use direct DB2 access. This file is the DB2 ODBC initialization file, which is read at run time to determine the default DB2 system ID and other DB2-specific parameters. The use of a DSNAOINI file is detailed later in this discussion.

Static SQL

When you create a Query class, you can choose that the query be *static* or *dynamic*. Since the Query Editor is only available in the Windows version of CA Aion BRE, query and field attribute-specific properties (such as alias name) can only be modified from the Windows version of CA Aion BRE.

Important! To use static SQL with CA Aion BRE, you must first allocate a PDS or PDSE named DBRM in which to store the DB2 resource definitions that are generated when the Aion application is compiled. The DBRM library must have the same high-level qualifier as the Aion installed files and libraries (as specified in the Aion IDE settings). This library must have attributes: fixed block, LRECL=80.

CA Aion BRE uses multi-row fetch by default. If your environment is not DB2 v8 New Function Mode (NFM), or DB2 v9, then multi-row fetch cannot be used for Static SQL requests. An SQL error -4700 will occur if a multi-row fetch is performed in an environment that does not support this capability.

To disable the use of multi-row fetch with Static SQL, in the BABUILD step SYSIN file specify the following new parameter:

```
DB2-STATIC-MULTIROW-FETCH=NO
```

It causes the generated Static SQL code to perform single row fetch activities instead. If the value of the DB2-STATIC-MULTIROW-FETCH parameter is any other value, or unspecified, then static SQL multi-row fetch capabilities will be generated.

When you build an application with static SQL, CA Aion BRE generates the standard C source, and an additional C program containing embedded SQL statements (in a *.SQC[STATCPP] data set). CA Aion BRE preprocesses to generate the C file and DBRM, and the C file is compiled and linked into the DLL.

You must consider the following constraints and factors when using static SQL:

- CA Aion BRE does not support interpretive execution of static SQL. If you interpretively run an application that contains static SQL, CA Aion BRE ignores the static property. Only built applications use static SQL at run time.
- At compile time, static SQL is only generated for Query classes that have the static SQL property checked and have a DB2 connection specified. When testing mainframe applications on the PC, you can use run time logic to temporarily switch a Query class connection pointer between DB2 and ODBC (or other SQL interface). If you wish to use static DB2 SQL on the mainframe, and another SQL interface to test under Windows, we recommend that you configure your application to use DB2 as the default edit-time connection type. Then swap to the Windows SQL interface, that is, ODBC at run time on Windows, as opposed to the other way around. CA Aion BRE can only generate static SQL from a DB2 connection.
- CA Aion BRE uses data types corresponding to the C type. Host variable allocation size should match the DB2 column definition. For example, a host variable (marker) for a CHAR(10) column should be defined in CA Aion BRE as string(10), rather than just string, which would default to 1024 bytes. For best performance, DB2 SMALLINT columns should be defined in CA Aion BRE as integer(2). If the host variable (marker) for a SMALLINT column does not specify a length of 2, the data type mismatch results in a DB2 table or index scan. For large tables, this can cause performance degradation.
- For static SQL, Aion Execution Trace output contains host variables instead of actual values.

- Null and unknown markers are not supported in static SQL. Unlike in dynamic SQL, the marker is always used in the WHERE clause in static SQL, regardless of whether its value is null. If a null value is detected in a marker for a static query, an error message is generated at run time.
- On UPDATE operations (including INSERT) involving static SQL, CA Aion BRE updates all non read-only mapped columns that are not read-only. This can have performance consequences, and can cause unexpected results related to column locks. You should mark all columns you do not intend to modify as read only.
- Whenever a row is inserted, any read-only or unmapped table columns default to a null value. Therefore, when inserting, all table columns defined as non-null values should be mapped as nonread only.
- Prior to connecting to DB2, the default plan for DB2 ODBC (DSNACLI) must have been installed on your system.
- Make the LE dynamic routines in SCEERUN and the DB2 load modules accessible through the STEPLIB DD card. For example:

```
//STEPLIB  DD DSN=CEE.SCEERUN,DISP=SHR         // LE dynamic
routines
//         DD DSN=SYS2.DB2.SDSNEXIT,DISP=SHR   // DB2 exit
modules
//         DD DSN=SYS2.DB2.SDSNLOAD,DISP=SHR   // DB2 load
modules
```

- Static SQL host variables must be simple data types. DB2 static SQL does not support LIST data-type host variables. (Using an Aion LIST marker for a static SQL query will result in generation of invalid C code in the SQL file containing the static SQL code.)

  A LIST marker is typically used for a WHERE clause that contains an IN statement.

  For example, consider a mapped marker MKNAME of type LIST OF STRING with an alias TABLE1.USERNAME. If the marker value is ('Smith','Jones'), the dynamic SQL generated at run time would be:

  ```
  SELECT... WHERE (TABLE1.USERNAME IN ('Smith', 'Jones')) ...
  ```

  To make this query valid for static SQL, you can use one of the following routines:

  – Unmap the marker (or remove the IN statement from the WHERE clause), and add a WhenFetched method to perform the IN statement logic. The WhenFetched method is invoked each time a row is fetched from a result set. Return TRUE to accept the row or FALSE to reject the row.

  – If you have a fixed number of elements in the list, use separate unmapped markers instead of a list. For example:

    ```
    WHERE ... IN (:mk1, :mk2, :mk3).
    ```

- For static SQL, DB2 is case-sensitive. Although most mainframe applications contain uppercase code exclusively, Aion applications, and SQL table and column names created on a PC frequently contain mixed or lowercase characters. Case discrepancies are allowed with dynamic SQL, but will cause SQL bind errors with static SQL. If using static SQL, ensure all SQL keywords entered in the Query class SELECT statement are specified in uppercase. Additionally, ensure all column and table names in the SELECT statement or alias names match the case as defined in the DB2 tables.

Bind Static DB2

When building an Aion application containing static SQL, you must explicitly bind the plan (this step is typically performed by a DBA). Since dynamic ODBC access to DB2 is required for static SQL queries, the generated DBRM must be bound into the existing plan for ODBC *planname*.

The default name for *planname* is DSNACLI. You can specify any name for *planname*. If you specify a name other than the default, the run time JCL for the application must include a DSNAOINI file that specifies the alternate plan name.

We recommend that you use the following steps when binding static SQL. For clarity, the default plan name DSNACLI is used.

To bind static DB2

1. Add a package list to the DSNACLI bind control cards.
2. Rebind the plan.

   Note: Aion packages can then be rebound as needed without rebinding the plan.

   All Aion packages should be grouped under one collection ID.

   The following example shows control cards binding an application named APPONE with collection ID AION:

   ```
   DSN SYSTEM(DB2A)
   BIND PACKAGE (AION) MEMBER(APPONE) ISOLATION(CS)
   LIB('dbrm-lib');
   END
   ```

   dbrm-lib

   The PDS containing the DBRM (as specified in the build job).

3. Add a package group specification to the bind plan control cards for DSNACLI.

   In the following example, the specification is the last line of the *.SDSNSAMP(DSNTIJCL) IBM-delivered control cards:

   ```
   DSN SYSTEM(DSN1)
   BIND PLAN(DSNACLI)
   PKLIST(DSNAOCLI.DSNCLICS
   DSNAOCLI.DSNCLINC
   DSNAOCLI.DSNCLIRR -
   DSNAOCLI.DSNCLIRS
   DSNAOCLI.DSNCLIUR
   DSNAOCLI.DSNCLIC1
   DSNAOCLI.DSNCLIC2
   DSNAOCLI.DSNCLIF4 -
   DSNAOCLI.DSNCLIMS
   DSNAOCLI.DSNCLIQR
   AION.*      <- add this line after the final 'DSN*' line
   END
   ```

   Note: For DB2 v8 New Function Mode and higher add package to package list as follows:

   DSNAOCLI.DSNCLINF

   Failure to bind DSNCLINF into a package and/or an application plan could result in SQLCODE -805 at application run time.

4. Once a single package in the AION group exists, rebind the DSNACLI plan one time. Thereafter, any number of applications can be bound and rebound in AION without having to rebind the DSNACLI plan.

More Information:

Using a DSNAOINI File (see page 56)

Using a DSNAOINI File

The DSNAOINI file contains optional DB2 ODBC connection parameters. Depending upon how DB2 ODBC was installed, the MVSDEFAULTSSID parameter may be required to establish a connection to the desired DB2 subsystem. If so, the MVSDEFAULTSSID parameter must be included in the [COMMON] section of the DSNAOINI file.

Whether the MVSDEFAULTSSID parameter was specified, the PLANNAME parameter must be specified if the default DB2 ODBC plan name DSNACLI is not used. The PLANNAME parameter must be included in the [ssid] section.

The following example illustrates how to properly specify these parameters:

```
//DSNAOINI DD *
    [COMMON]
    MVSDEFAULTSSID=DB2A
    [DB2A]
    PLANNAME=AONACLI
/*
```

To obtain DB2 ODBC traces for general application debugging and diagnosis, enable the APPLTRACE and APPLTRACEFILENAME keywords in the DB2 ODBC initialization file.

For more information, see see the appropriate IBM UBD for z/OS ODBC guide.

Using an AONAOINI File

You can use an AONAOINI file dynamically to specify DB2 connection parameters. The DB2 ODBC connection parameters are normally static and are specified in a DSNAOINI data set. The contents of the AONAOINI file look exactly like the contents of a DSNAOINI file (with one exception, which is explained later). However, if the execution JCL includes a DD for AONAOINI instead of DSNAOINI, Aion applications can use the ConnnectInfo attribute on the DB2 connection instance to specify dynamically any desired DB2 ODBC connection parameters. If the ConnectInfo attribute of the DB2 connection instance is not null (at the time the Connect method is invoked), its contents override the contents of the AONAOINI file. Values within of the ConnectInfo and the AONAOINI file are dynamically merged to compose DSNAOINI file contents. DB2 accesses the connection information in the dynamically merged DSNAOINI file.

Important! If the ConnectInfo attribute of the DB2 connection instance is null when the Connect method is invoked, the AONAOINI file is ignored and the connection is attempted using the installation's default parameter values. If an AONAOINI file is specified and your application does not need to override any AONAOINI file parameters, you can specify any DB2 ODBC connection parameter value (for example, APPLTRACE=0) in the ConnectInfo attribute of your application's DB2 connection instance to activate the AONAOINI facility. This ensures that the contents of the AONAOINI file are used to establish the DB2 ODBC connection.

The ability to dynamically specify any DB2 ODBC connection parameter is valuable. However, there may be circumstances when you need to enforce some measure of control over which parameters can be modified. To that end, only those parameters specified in the AONAOINI file are subject to override. However, the AONAOINI file must contain all the DB2 ODBC connection parameters that are to be dynamically merged, and so an additional means is required to control which of the parameters can be overridden.

For example, you may wish to preclude the overriding of the MVSDEFAULTSSID parameter. This would ensure that applications could only connect to the DB2 subsystem specified in the AONAOINI file's MVSDEFAULTSSID parameter. Similarly, an installation might want to restrict applications from dynamically specifying the PLANNAME parameter. Any combination of DB2 ODBC connection parameters can be excluded from override by listing them in the AONAOINI file in an optional section named IGNOREAPPVALUES. Like all other section names, it must be enclosed by square brackets [ ]. If present, the IGNOREAPPVALUES section must be the first section in the AONAOINI file. The connection parameters to be ignored are included immediately after the [IGNOREAPPVALUES] statement, specifying one connection parameter per line. The list of ignored parameters is delimited by the start of the COMMON section.

For example, the following AONAOINI file would ensure that all applications connect to the DB2A subsystem. The default DB2 plan name would be AONACLI, but each application could specify any DB2 plan name known to the DB2A subsystem:

```
//AONAOINI DD *
    [IGNOREAPPVALUES]
    MVSDEFAULTSSID
    [COMMON]
    MVSDEFAULTSSID=DB2A
    [DB2A]
    PLANNAME=AONACLI
/*
```

This mechanism works the same whether the Aion application is executed in batch or in MAES. However, due to the multitasking nature of MAES, there are other considerations that would generally not be an issue for a batch application. For example, assume the following:

- Some MAES applications will override the installation's default DB2 subsystem (DB2A, in this example) and connect to DB2 subsystem DB2B or DB2C.
- Certain MAES applications will need to override the default DB2 ODBC plan name of the DB2 subsystem.
- Establish the default plan names for subsystems DB2A, DB2B, and DB2C to be DB2ACLI, DB2BCLI, and DB2CCLI, respectively.
- Certain MAES applications could be in test phase and may need to have the DB2 ODBC trace facility activated (APPLTRACE=1), with the trace directed to a specific trace output data set (APPLTRACEFILENAME=dsn).

The AONAOINI file could be coded as follows:

```
//AONAOINI DD *
    [COMMON]
    MVSDEFAULTSSID=DB2A
    APPLTRACEFILENAME=DD:TRACEOUT
    APPLTRACE=0
    [DB2A]
    PLANNAME=DB2ACLI
    [DB2B]
    PLANNAME=DB2BCLI
    [DB2C]
    PLANNAME=DB2CCLI
/*
```

Suppose application TestOne needs to connect to the DB2A subsystem using DB2 plan TESTONE with a DB2 ODBC trace. While DB2A is the default DB2 subsystem, the default DB2 plan name for subsystem DB2A is DB2ACLI, and its APPLTRACE default value of 0 disables the DB2 ODBC trace. Since abends can result when multiple applications use the same DB2 ODBC trace file, it is best to override the default APPLTRACEFILENAME. In order to obtain the desired connectivity, application TestOne needs to specify DB2 ODBC connection overrides for PLANNAME, APPLTRACE and APPLTRACEFILENAME in the ConnectInfo parameter of its DB2 connection instance before the Connect method is executed (either explicitly or implicitly). The relevant code in TestOne might look as follows:

```
DB2Connection.ConnectInfo = “PLANNAME=TESTONE;APPLTRACE=1;
    APPLTRACEFILENAME='TSOID.CLITRACE.TESTONE'"
DB2Connection.Connect(false)
```

Further, suppose application TestTwo needs to connect to the DB2B subsystem using DB2 plan DB2BCLI without an DB2 ODBC trace. While DB2BCLI is the default DB2 plan name for subsystem DB2B, the default subsystem is DB2A. The default DB2 ODBC trace options are acceptable without modification. In order to obtain the desired connectivity, application TestTwo needs to specify an DB2 ODBC connection override for MVSDEFAULTSSID in the ConnectInfo parameter of its DB2 connection instance before the Connect method is executed (either explicitly or implicitly). The relevant code in TestTwo might look like this:

```
DB2Connection.ConnectInfo = "MVSDEFAULTSSID=DB2B"
DB2Query.Load()
```

At times, it may be desirable to verify the execution of the AONAOINI facility. If an AONAOINI DD statement is included in the execution JCL, and the application trace facility is enabled, an execution log is generated. This log presents the contents of the AONAOINI file, the application's ConnectInfo parameter value, and the resultant equivalent DSNAOINI file contents. The application trace records will look something like the following:

```
0.0560        ::Activating AONAOINI Facility
0.0610        ::AONAOINI Rec =   [COMMON]
0.0620        ::AONAOINI Rec =     MVSDEFAULTSSID=DB2Z
0.0620        ::AONAOINI Rec =     APPLTRACE=1
0.0620        ::AONAOINI Rec =     APPLTRACEFILENAME=DD:TRACEOUTZ
0.0620        ::AONAOINI Rec =   [DB2Z]
0.0620        ::AONAOINI Rec =     PLANNAME=AONACLZ
0.0650       ::App ConnInfo = PLANNAME=AONACLI;MVSDEFAULTSSID=DB2A
0.0670        ::DSNAOINI Rec =   [COMMON]
0.0680        ::DSNAOINI Rec = MVSDEFAULTSSID=DB2A
0.0680        ::DSNAOINI Rec =     APPLTRACE=1
0.0680        ::DSNAOINI Rec =     APPLTRACEFILENAME=DD:TRACEOUTA
0.0680        ::DSNAOINI Rec =   [DB2A]
0.0680        ::DSNAOINI Rec = PLANNAME=AONACLI
0.0700        ::Deactivating AONAOINI Facility
```

In reviewing traces of the AONAOINI facility, there are two points to keep in mind. First, depending upon how you view the trace output, the square brackets around the section names (e.g. [COMMON]) may not be visible. Second, it is common practice to indent the contents of the DSNAOINI file. If the same practice is followed with the AONAOINI file, it is easy to determine the source of the merged (DSNAOINI Rec) parameter values. Those values from the AONAOINI file retain their indentation while those provided by the application are not indented. This is seen in the immediately prior example of an AONAOINI facility trace.

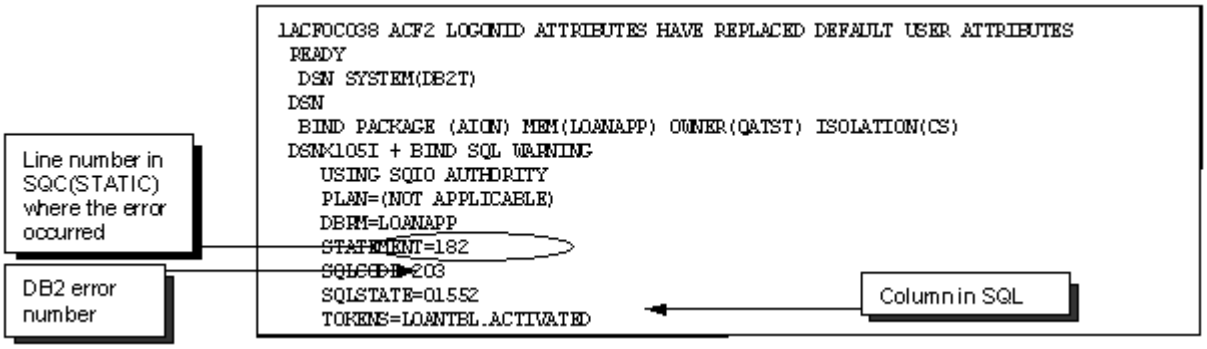
More Information:

Using a DSNAOINI File (see page 56)

DB2 Bind Errors

If a static SQL bind fails, the bind output contains DB2 errors, as well as line numbers which indicate the location (in the STATIC member of the *<aionhlq>.appname*.SQC data set) of the problem SQL statement. Use this information to modify the Query class, which handles the associated SQL statement in your application. See the IBM DB2 manuals for error code explanations and suggestions on solving the errors.

For example, consider the bind output with errors shown in the following example:



The circled line in the preceding example indicates which line of the STATIC member in the SQC data set contains the offending SQL statement (in this case, line 182).

XML

CA Aion BRE includes capabilities for processing Extensible Markup Language (XML) content in text files, or text string values (which could be relational database column values). XML text provides information that is easily read by people and computers. The ability of Aion BRE to process XML content will be very helpful for you. XML is one of the most important developments in document syntax within the history of computing.

There are two fundamental methods of processing XML content:

1. DOM : document object model
2. SAX : the simple API for XML

XML examples are found in <aionhlq>.SAMPLES.JCL.XML and instructions on how to use these examples are found in <aionhlq>.SAMPLES.README(XML) file.

Note: Aion BRE XML capabilities are optionally installed. If these capabilities are absent, please request installation of XML capabilities from your installation administrator.

Prerequisite

Aion BRE XML components use underlying capabilities provided by:

- IBM's XML Toolkit for z/OS. IBM's XML toolkit is an implementation of Apache Xerces-c capabilities.
- The z/OS XML Toolkit must be obtained separately from IBM.

References

The following publication describes the z/OS XML toolkit:

- "XML Toolkit for z/OS User's Guide" (IBM Publication: SA22-7932-05)
- The following publication is another helpful z/OS XML reference:
  - "Using XML on zOS and OS390 for Application Integration" http://www.redbooks.ibm.com/redbooks/pdfs/sg246285.pdf (IBM Publication: SG24-6285-00)

DOM Processing Considerations

When an XML file is processed via DOM methods, the entire file is loaded into memory in a tree structure. This provides opportunities for referencing XML content within different document sections at the same time.

Very large XML files can be difficult to process using DOM methods, as a consequence of the amount of memory required.

DOM is mandatory and the only XML component that Aion BRE can use to UPDATE and/or CREATE XML document.

No information is available until the entire XML file has been processed. An out of storage condition can occur before the entire file has been processed.

SAX Reader Processing Considerations

When an XML file is processed via SAX methods, callback methods receive control when different document events occur. For example, methods are invoked when the following events occur:

1. The XML document is initially being processed.
2. An XML begin element is being processed.
3. The textual content associated with an element is complete.
4. An XML element is complete.
5. The XML document is complete.

Unlike DOM, memory is only required for a section of the XML content.

It is not possible to reference XML content that is in another section.

Very large XML files can be easily processed using SAX methods, as minimal memory is required.

SAX can work only in read mode.

SAX reader processing can conclude before the whole XML file is processed. For example, when the content of a specific element matches search criteria.

DOM Against SAX Reader Processing Considerations

Whether to process an XML file using DOM or SAX methods is a decision that must be weighed when an application is initially designed. If it is known that the application might process very large XML files, then using SAX methods may be the only viable alternative. On the other hand, if it is known that the application will process moderately sized XML files, then either DOM or SAX methods can be used. In this case, either the simpler SAX methods will suffice, or DOM methods can be used if XML content that is in different sections need to be referenced at the same time.

DOM is mandatory and the only XML component that Aion BRE can use to UPDATE and/or CREATE XML document. Where SAX can work only in read mode.

Note: SAX methods cannot be used to update XML files.

Supporting Application Libraries

Aion BRE XML capabilities are supported by the following libraries:

- DOMLIB.APP - external DOM processing methods
- SAXLIB.APP - SAX processing methods
- XMLLIB.APP - internal DOM processing methods (used by DOMLIB.APP).
- The DOMLIB and SAXLIB libraries are intricate, consequently the individual methods within these libraries will not be described here. Some example files are provided to help you with typical XML activities.

XML Example Files

Example files are provided to help you use Aion BRE XML capabilities. There are two examples: DOMRUNF and SAXRUNF. Both examples reference XML content that is in file: <aionhlq>.SAMPLES.DATA

The source of the examples can be found in: <aionhlq>.SAMPLES.APPS.

The XML example applications should be extracted to a files named DOMRUNF.APP and SAXRUNF.APP. The applications must be allocated with the standard Aion application source attributes: DSORG(PS) RECFM(VB) LRECL(1024) BLKSIZE(27998).

Note: These examples are extracted as part of the post installation step, APPCOPY.

DOMRUNF : DOM Example Application

The DOMRUNF application processes the XML file using DOM methods. This program references the XML file via: dd:xmlsamp. Consequently, the XML file is allocated in the XMLSAMP file in the DOMRUN*.jcl jobs.

SAXRUNF : SAX Example Application

The SAXRUNF application processes the XML file using SAX methods. This program references the XML file via: dd:xmlsamp. Consequently, the XML file is allocated in the XMLSAMP file in the SAXRUN*.jcl jobs.

Batch Jobs to Build the XML Examples

The example jobs to build the XML examples can be found in: <aionhlq>SAMPLES.JCL.XML.

DOMBLD : Batch Build DOM Application

The DOMBLD job builds the DOMRUNF.APP for non-XPLINK execution.

DOMBLDX : Batch Build DOM Application, XPLINK

The DOMBLDX job builds the DOMRUNF.APP for XPLINK execution.

SAXBLD : Batch Build SAX Application

The SAXBLD job builds the SAXRUNF.APP for non-XPLINK execution.

SAXBLDX : Batch Build SAX Application, XPLINK

The SAXBLDX job builds the SAXRUNF.APP for XPLINK execution.

Batch Jobs to Execute the XML Examples

The jobs that execute the XML examples are members of: <aionhlq>SAMPLES.JCL.XML.

DOMRUN : Process XML Using DOM Capabilities

The DOMRUN job runs the DOMRUNF.APP via non-XPLINK execution.

DOMRUNX : process XML using DOM capabilities, XPLINK

The DOMRUNX job runs the DOMRUNF.APP via XPLINK execution.

SAXRUN : Process XML Using SAX Capabilities

The SAXRUN job runs the SAXRUNF.APP via non-XPLINK execution.

SAXRUNX : Process XML Using SAX Capabilities, XPLINK

The SAXRUNX job runs the SAXRUNF.APP via XPLINK execution.

MQSeries Examples

CA Aion BRE also provides examples which demonstrate the use of MQLib, for accessing IBM's MQSeries messaging middleware. All commonly needed functionality is shown in the first example, MQBasic. The other examples demonstrate the less common features:

MQBasic

Demonstrates all the basic functionality typically needed for getting and putting messages to and from MQSeries.

MQInq

Available only in MQSeries for MVS, it performs an Inquire on a process.

MQPut1

Compares the use of MQPut with MQPut1, to help in determining if MQPut1 would be useful.

MQRecs

Demonstrates the usage of all the less common MQSeries control blocks and other structured records not already shown, such as the Transmission Queue Header.

Information regarding these examples can be found in file <aionhlq>.SAMPLES.README(MQSERIES).

Chapter 3: Build and Manage Aion Applications

This chapter describes how to customize, build, and manage your Aion applications.

This section contains the following topics:

Create Applications Using Windows-based Remote Development

Aion applications for the mainframe and UNIX platforms should be developed using the Windows-based IDE. CA Aion BRE requires you to create a *remote* application on a Windows-based workstation and transfer it to the mainframe. You can use Remote Development to build applications intended for the mainframe and UNIX platforms.

Remote Development uses FTP to Save and Restore Aion application files to and from the mainframe. It can also generate and submit JCL (again using FTP) to run a Batch Build job that builds the application on the mainframe. Job output is displayed in the Windows-based IDE.

The following list provides a brief description of some of the Remote Development features.

Remote Settings

This Windows dialog maintains all data stored in the registry.

The settings are not unique to one application; you must set wrappers as required for each application

Remote Restore

This Windows dialog copies an application from the mainframe to Windows, restores it, and opens it.

The application is then just like any other Windows application.

Remote Save

This Windows dialog performs a normal save, rewriting the local application file. It then copies the application from Windows to the remote mainframe in preparation for Remote Build.

Any application may be remotely saved, regardless of whether it was remotely restored.

Remote Build

This Windows dialog submits a batch build job to the remote platform to build the remote version of an application using Batch Build.

Job output is returned to Windows for analysis. The current remote version of the application is used, as last created by Remote Save.

The first step is always a restore, creating the .BIN file, which Remote Run may then use.

Additional steps are present (based on the WRAPPER= parameter setting) to build the application. The application is then compiled. If it is a batch app, it can be executed either using Remote Submit, or directly in the remote environment

Remote Run

This Windows dialog submits a batch job invoking REEXEC to interpretively execute the remote copy of an application.

Job output is returned to the user for analysis. The current remote version of the .BIN file is used, as last created by Remote Restore or Batch Restore.

Remote Submit

This Windows dialog allows any user-supplied JCL to be submitted to the remote Z/OS platform. Thus, any desired non-automated procedures can be executed from the PC. Typical uses include: DB2 binds, PL/I or COBOL compiles, and executing the BAALOCAT and BAALOCLI procedures when the default space allocations are not sufficient. Alternatively, the distributed JCL procedures can be customized in a user procedure library that is specified in the JCLLIB ORDER card.

Using the Windows-based features listed previously and Batch Build, you perform the following general procedure to create and build an Aion application.

To create an application using Window-based Remote Development

1. Write and debug the application using the Windows-based IDE.
2. From Windows, save the application to the remote platform.
3. Use Remote Development Build or Batch Build to build your application on the remote system.
4. Review the output to ensure that the compilation completed successfully.

More information on application development is provided in the *CA Aion BRE Product Guide.*

Build an Aion Application

To use Remote Development on your Windows workstation

1. To configure your CA Aion BRE workstation for Remote Development, select File, Remote Development, Enable Remote Development.
2. Select File, Remote Development, Settings.

   The Remote Settings dialog displays.
3. On the General tab, use the Path textbox to specify the location of your mainframe application files.
4. On the Data Definition tab, set your Space requirements. LRECL and BLKSIZE can be left blank.
5. On the Data Management tab, define whatever classes are defined for your site.
6. On the Job Control tab, define your Job cards and Batch Build parameters.

The Job cards section allows the specification of JCL cards related to the entire job execution, such as Job Statement and JCLLIB ORDER cards. The JCLLIB ORDER card may be used to override some or all of the Aion Batch Build Procedures with user-customized JCL procedures.

Parameter	Value
Application high-level qualifier	APPHLQ=tsoid.hlqid
Application name	APP=appname (do not include .app suffix)
Wrapper	[WRAPPER=]"DRIVER" "C", "C-CICS", "C-IMS", "C++"

Parameter	Value
	"COBOL", "COBOL-CICS", "COBOL-IMS" "PL1", "PL1-CICS", "PL1-IMS", "TCPIP", "RESTORE-ONLY" [or "NONE"]
List of high-level qualifiers for included system or application libraries (.app and .bin files)	LIB=HLQ1,HLQ2… **Note:** The last high-level qualifier in the list is also used as the value of the AIONHLQ symbolic parameter in the Aion Batch Build procedures to specify the location of the product load library (for example, AIONHLQ.LOAD

Optional parameters include:

Parameter	Value
Trace flag	TRACE
Embedded interpreter	EMBEDDED-INTERPRETER
Component trace	COMPONENT-TRACE
Indirect CICS static SQL	CICS-STATIC-SQL
To use the high-performance, manual suspend/resume protocol for IMS or CICS clients (rather than the automatic suspend/resume protocol)	CONVERSATIONAL
Build and execute using the high-performance XPLINK option	XPLINK
Generate a single row fetch syntax for DB2 only	DB2-STATIC-MULTIROW-FETCH=NO

Important! On mainframe platforms, CA Aion BRE does not recursively build included libraries. Therefore, prior to building an application, you should individually compile all included user libraries (not Aion-supplied libraries such as SysLib or DataLib), and copy the generated DLL files to the appropriate load libraries. This ensures that built applications always use current versions of included user libraries.

Use a standard Z/OS batch environment to build your application. Execute the REBBnn Batch Build program to invoke the Aion Build services.

When the BABUILD procedure is executed on the mainframe, the build places generated C code in temporary data sets that can be compiled and linked in subsequent job steps. Remote development automatically generates the required job steps to compile the generated C code and link the Aion component. In addition, for stand-alone applications, a driver program to execute the Aion component will be generated, compiled and linked.

When you deploy Aion applications as batch programs, you use them as stand-alone applications or as embedded applications in another program. To deploy a stand-alone batch program, build the application with WRAPPER="DRIVER", and execute $APPNAME. For use as an embedded application, specify WRAPPER= as appropriate in the Remote Development settings and Batch Build parameters.

Specify Execution Trace Output

While running an Aion application, you can generate Execution Trace output to a data set or to the MAES log. Execution Trace provides output regarding the status and actions of an Aion application executing in the mainframe environment. You use Execution Trace output to debug application logic.

Information contained in the Execution Trace output includes:

- Elapsed CPU time since execution start
- SQL execution
- Rule processing
- Libraries referenced by the application, including whether they are running in interpreted or compiled mode
- The names of executing methods
- Value assignments that occur during method execution

Important! Generating Execution Trace output can affect performance of your application execution. Generally, applications should only be built to generate Execution Trace output during testing phases.

You can write your own statements to the Execution Trace output for both interpreted and compiled execution using the WriteToTrace method. For example the following statement writes the following line in the Execution Trace output:

```
WriteToTrace("Age="$FORMAT(p.age))
**Age=36
```

The following figure shows an example of Execution Trace output:

```
0.00    0.0000  Aion 11.0, Build Mar  9 2009 (z/OS MAES):: 2009/03/09 15:22:19
0.00    Events to Trace: All events
0.00    -CHKORDER (interpreted)
0.00    -syslib (compiled)
0.00    -iolib (compiled)
0.00    -executing Start (Main)
0.01      Main.reportfile="DECTABLE.RPT"
0.01      pfile=File_00001
0.05      p=ConfigMachine_00001
0.06      -executing INFERConfiguration (ConfigMachine_00001)
0.06        bFwd=TRUE
0.06        -INFER BEGIN
0.06          -executing DecideApproved (ConfigMachine_00001)
0.06            -rule posted: "Determine Approved", priority: -5
0.07          -completed DecideApproved
0.07          -executing DecideOperatingSystem (ConfigMachine_00001)
0.07            -rule posted: "Determine OS", priority: -10
0.07          -completed DecideOperatingSystem
0.07          -executing DecidePrice (ConfigMachine_00001)
0.07            -rule posted: "Determine Price", priority: -10
0.07          -completed DecidePrice
0.07          ConfigMachine_00001.SoftwareCompleted=TRUE
0.07          -Forwardchain begin
0.07            -begin premise evaluation: "Determine Approved"
0.08            _Condition_1="Manager"
0.08            -executing GetCurrentMachineCount (ConfigMachine_00001)
0.08            -completed GetCurrentMachineCount
0.08            _Condition_3=0
0.08            -rule pended: "Determine Approved", pended for:
ConfigMachine_00001
0.08            -begin premise evaluation: "Determine OS"
0.08            _Condition_1="Marketing"
```

Execution Trace Output Processing

Trace Destination: Dynamic File Against AIONTRAC DD

For both *interpreted* and *compiled* executions, the Execution Trace output is written to a dynamically allocated dataset, unless redirected using an AIONTRAC DD allocation. For consistency and flexibility, this redirection capability is available in all execution environments. However, additional considerations apply in the MAES environment.

Under MAES, S02D abends can terminate the MAES region if tracing activity is attempted from concurrently executing applications. For that reason, the AIONTRAC DD is not included in the distributed MAES JCL. The use of AIONTRAC with MAES is very convenient, but it is only appropriate for single-threaded validation or testing situations.

Applications: Interpreted Against Compiled

For *interpreted* executions, Execution Trace output is always generated for the Application, unless you turn the Execution Trace off using TraceEnable in the _utilities class. Execution Trace output from *interpreted* executions contains more information than Execution Trace output from *compiled* executions. For example, only interpreted Execution Trace output includes data assignments. For *interpreted* executions, the name of the dynamically allocated Execution Trace file has the form *APPHLQ.APPNAME*.TRACE.

For *compiled* executions, Execution Trace output is only generated if the TRACE or COMPONENT TRACE option is specified in the BABUILD JCL procedure. The COMPONENT TRACE option is only applicable to applications being executed under MAES.

When the TRACE option is specified, the Execution Trace output is written to the data set specified by the AIONTRAC DD or to a dynamically allocated data set. For *compiled* executions, the name of the dynamically allocated Execution Trace file has the form *JOBUSERID.APPNAME*.TRACE (for batch execution) or *JOBUSERID.ONLINEUSERID.APPNAME*.TRACE (for execution under MAES). If MAES is executed as a started task, the value for *JOBUSERID* is the ACID with which the started task is associated in RACF. Using the default ACID, STCDEF, usually results in security violations, and is not recommended.

When the COMPONENT TRACE option is specified, the Execution Trace output is written to the MAES log (DIV file). MAES component execution trace output can be viewed during MAES execution (using the MAES Log viewer) or after MAES execution (using the DLOG utility program).

Compiled applications outside of MAES only generate Execution Trace output if executed from the application driver (the generated load module named after the application, except prefixed with a '$').

More Information:

View the MAES Log (see page 213)

Copy Log File Archives (see page 216)

Build an Application Using Batch Build

Using Batch Build in a standard mainframe environment with its simple compile and link procedures allows you to build Aion applications through a JCL. Sample JCL for constructing the build job stream is distributed with the Aion product in <AIONHLQ>.SAMPLES.JCL.BATCHBLD. You can tailor this JCL to your particular environment. See member BATCHBLD in the <AIONHLQ>.SAMPLES.README for an explanation of the Batch Build procedures and the Batch Build JCL samples.

Execute procedure BABUILD to perform the build under the control of parameters you specify in SYSIN. The build places generated C code in the appropriate data sets so that it can be compiled and linked in subsequent job steps. No JCL is created or submitted asynchronously. Separate compile and link steps must be executed.

CA Aion BRE supplies Batch Build procedures you can use when building your applications.

More Information:

Batch Build Procedures (see page 82)

Specify Parameters for Batch Build

Specify Batch Build parameters by supplying values for keywords. The following parameters are required.

APPHLQ=hlq

Application high-level qualifier-Specify all of the Z/OS data set name nodes that precede the actual application name

APP=appname

Application name-Specify the name of the application.

The Z/OS data set name is constructed by concatenating APPHLQ and APP, and appending the extension .APP. Thus, APPHLQ=USERID1.AION APP=HELLO would utilize file USERID1.AION.HELLO.APP.

WRAPPER=type

Wrapper type-This corresponds to the Aion Component Build Directives panel in conjunction with the Language and System options you supply in the Aion MAES Component Settings panel

Specify one of the following types for this parameter:

DRIVER

Generates both an executable driver program and the DLL representing the application.

C

Wraps the DLL for invocation from a C program running in batch or TSO

C-CICS

Wraps the DLL for invocation from a C program running in CICS.

C-IMS

Wraps for a C program running in IMS-DC.

TCPIP

Wraps for CICS TCP/IP

C++

Wraps for C++ running in batch or TSO.

COBOL

Wraps for COBOL batch or TSO

COBOL-CICS

Wraps for COBOL in CICS.

COBOL-IMS

Wraps for COBOL in IMS-DC.

PLI (or PL1)

Wraps for PL1 in batch or TSO.

PLI-CICS (or PL1-CICS)

Wraps for PL/I in CICS.

PLI-IMS (or PL1-IMS

Wraps for PL/I in IMS.

RESTORE-ONLY

Causes the application to be restored, but no code is generated from it. As this value is an alternative to a real wrapper, it can also be specified without prepending WRAPPER=

LIB=

LIB=HLQ1,HLQ2…

The list of high-level qualifiers for included system and application libraries (.app and .bin files).

The following parameters are optional:

TRACE

Generate an Execution Trace file.

EMBEDDED-INTERPRETER

Build for testing purposes through the embedded interpreter.

COMPONENT-TRACE

Build for testing purposes using component trace statements.

CICS-STATIC-SQL

Utilize indirect static SQL under CICS control for all static queries.

CONVERSATIONAL

Suppress Suspend/Resume logic when invoked from a CICS or IMS program.

XPLINK

Build using the high-performance XPLINK compiler option.

DB2-STATIC-MULTIROW-FETCH=NO

Generate a single row fetch syntax for DB2 only.

Important! CA Aion BRE uses multi-row fetch by default. If your environment is not DB2 v8 New Function Mode (NFM), or DB2 v9, then multi-row fetch cannot be used for Static SQL requests. An SQL error -4700 will occur if a multi-row fetch is performed in an environment that does not support this capability.

To disable the use of multi-row fetch with Static SQL, in the BABUILD step SYSIN file specify the following new parameter:

```
DB2-STATIC-MULTIROW-FETCH=NO
```

It causes the generated Static SQL code to perform single row fetch activities instead. If the value of the DB2-STATIC-MULTIROW-FETCH parameter is any other value, or unspecified, then static SQL multi-row fetch capabilities will be generated.

Separate parameters in a string from each other by a space. Parameters can occupy up to 10 lines, but no one parameter can span across multiple lines. Any order is acceptable. If repeated, a parameter's last value is used. Unrecognized text is ignored; this allows you to use noise words (such as NOTRACE) to document your application. Here is a sample parameter specification:

```
//SYSIN DD *
APPHLQ=USERID1.AION
APP=HELLO
LIB=LIBHLQ
WRAPPER=C-CICS
TRACE
/*
```

Batch Build Procedures

A set of JCL procedures is provided in library <aionhlq>.PROCLIB to assist you in keeping jobs simple. All procedures described in the following table begin with the letters BA, indicating batch. Some of the procedures have a name ending with the letter X. These procedures are used to build XPLINK applications.

The following list describes the JCL procedures.

BAALOCAT

Pre-allocates application-generated common files (only required for large applications).

BAALOCLI

Pre-allocates application-generated C client files (only required for large applications).

BABUILD

Executes Batch Build to prepare applications for non-XPLINK execution. It is designed for SYSIN to be overridden.

BABUILDX

Executes XPLINK Batch Build.

It is designed for SYSIN to be overridden.

BACMPCLI

Performs client-side compile for CICS and IMS wrapped applications.

BACOMP

Performs simple, or server-side (MAES) compile.

BACOMPX

Performs simple, or server-side (MAES) compile with the XPLINK option.

BADELCLI

Deletes client-side work files.

BADELETE

Deletes simple or server-side work files.

BADELSTA

Deletes work files used for static SQL.

BALINK

Performs simple or server-side (MAES) link.

BALINKX

Performs simple or server-side (MAES) link with the XPLINK option

BALNKCLI

Performs client-side link.

BALNKDRV

Performs link on driver program

BALNKDRX

Performs link on driver program with the XPLINK option.

BARUN

Executes an Aion application interpretively

BARUNX

Executes an Aion application interpretively, with support for XPLINK compiled user libraries.

BASTATIC

Performs a DB2 pre-process on static SQL module.

CICSDB2

Binder include statements for building a CICS DB2 client.

COMPDB2

Binder include statements for building a DB2 client.

DIRECT

JCL directives for clients that access DB2 in the same address space.

INDIRECT

JCL directives for clients that access DB2 in another address space.

MAES

Procedure for executing MAES.

NODB2

Binder include statements for building a non-DB2 client.

NONE

JCL directives for non-DB2 clients.

All generated C code is stored in two data sets, one having a suffix of .C (used to build the application component DLL), and the other (only generated for IMS and CICS C clients or for CICS PL/1 and COBOL clients using the CICS-STATIC-SQL build option) having a suffix of .C2 (used to build the client DLL). A compile of all members is performed using a single step, by *not* specifying the member name. If a driver program is requested, it is stored in the .C data set and compiled by the single compile step. However, it requires a separate link step, BALNKDRV.

Static SQL modules containing SQL statements are stored in a separate data set having the suffix .SQC. By default, static SQL access is performed from the Aion component DLL. However, if the CICS-STATIC-SQL build option is specified, static SQL access is performed from the CICS client DLL. After DB2 preprocessing, the static SQL source module is copied into the .C file (if static SQL access is performed from the Aion component DLL) or the .C2 file (if static SQL access is performed from the CICS client DLL). Since the BASTATIC procedure updates source files, it must precede the BACOMP and BACMPCLI procedures in the execution JCL.

The .C, .C2, and .SQL files are not the only files generated by the application build process. There are two files which are always generated along with the .C file (.BIN and .BUILD) and two files which are always generated along with the .C2 file (.BIN2 and .BUILD2). In addition, for COBOL and PL/1 clients, another file will be generated (.COBOL or .PL1). For large applications, these generated files may need to be pre-allocated to avoid exceeding their default capacities. The build procedures BAALOCAT and BAALOCLI can be used to pre-allocate larger files for use by the application build process.

All Batch Build procedures are required to specify the AIONHLQ symbolic parameter, as follows:

```
AIONHLQ=prodhlq
```

where *prodhlq* is the high-level qualifier of the product load library (for example, PRODHLQ.LOAD).

The following example builds and executes an application named HELLO:

```
//HELLODRV JOB (10223),CXC,CLASS=A,MSGLEVEL=(1,1),MSGCLASS=X
//*
//*********************************************************************
//*
//*  SAMPLE JCL TO BUILD AND RUN HELLO APP IN BATCH VIA REBB.
//*
//*  THE HELLODRV SAMPLE SUPPORTS THE FOLLOWING WRAPPER:
//*    DRIVER  <BUILD DIRECTIVE #1>
//*
//*  C CODE IS GENERATED BY CALLING PROC BABUILD, WHICH EXECUTES
//*  PROGRAM REBBnn. (For r11.o the module name is REBBBO.)
//*   THE APP AND DRIVER ARE BOTH COMPILED VIA A SINGLE
//*  INVOCATION OF BACOMP. THE APP IS LINKED VIA BALINK, WHILE THE
//*  DRIVER IS LINKED WITH BALNKDRV. WORK FILES ARE DELETED VIA BADELETE.
//*  FINALLY, THE APP IS RUN VIA IT'S DRIVER.
//*
//*
//*  Note: THAT PROC BALNKDRV DOES NOTHING IF CC>=6, THE VALUE RETURNED
//*  BY BABUILD IF WRAPPER=DRIVER IS REQUESTED BUT THE APP HAS NO
//*  'START' METHOD.
//*
//*  IT IS NORMAL TO PLACE BOTH APP AND DRIVER IN THE SAME LIB.
//*
//*********************************************************************
//*
//      JCLLIB ORDER=(CPABRE.PROD.PROCLIB)
//*
//*---------------------------------
//*  BUILD APP, PRODUCING CODE IN *.C, LINK CONTROL CARDS IN *.BUILD.
//*---------------------------------
//S01BUIL EXEC BABUILD,AIONHLQ=PDAION.PROD
//PS1GEN.SYSIN DD *
    APP=HELLO
    APPHLQ=USERID1.AION
    WRAPPER=DRIVER
    LIB=PDAION.PROD

/*
//*
```

```
//*---------------------------------
//*  COMPILE CODE (INCLUDING DRIVER)
//*  STORE OBJ'S IN *.BUILD
//*---------------------------------
//S02COMP EXEC BACOMP,
//             APP=HELLO,
//             APPHLQ=USERID1.AION,
//             AIONHLQ=PDAION.PROD
//*
//*---------------------------------
//*  LINK APP AND DRIVER
//*---------------------------------
//S03LKAP EXEC BALINK,
//             APP=HELLO,
//             APPHLQ=USERID1.AION,
//             AIONHLQ=PDAION.PROD,
//             DLLLIB=USERID1.AION.LOAD
//*
//S04LKDR EXEC BALNKDRV,
//             APP=HELLO,
//             APPHLQ=USERID1.AION,
//             AIONHLQ=PDAION.PROD,
//             LOADLIB=USERID1.AION.LOAD
//*
//*---------------------------------
//*  DELETE WORK PDS'S
//*---------------------------------
//S05DEL  EXEC BADELETE,
//             APP=HELLO,
//             APPHLQ=USERID1.AION
//
//*
//*---------------------------------
//*  EXECUTE THE APPLICATION VIA THE DRIVERDELETE WORK PDS'S
//*---------------------------------
//S06DEL  EXEC PGM=$HELLO,COND=(6,LE)
//STEPLIB   DD DSN=USERID1.AION.LOAD,DISP=SHR
//          DD DSN=PDAION.PROD.LOAD,DISP=SHR
//SYSPRINT  DD SYSOUT=*
//
```

More Information:

Build an XPLINK Application (see page 88)

Interpret the Results

Batch Build sets a precise condition code so that subsequent compile, link, and other steps do not need to be executed inappropriately. Following are the return codes and their meanings:

Category	Return Code and Description
Success Code	**0**-OK. Proceed with compile/link.
Warning Codes	**4**-Illogical parameter combination. A CICS or IMS-only option was requested without a CICS or IMS wrapper. Batch Build has successfully completed, ignoring the option(s). Proceed with compile/link. **5**-Batch Build has successfully completed, but invalid objects were detected during the restore process. Compiles and links should be suppressed. **6**-Code was generated successfully for the application, but the requested driver program was not built, despite the WRAPPER=DRIVER specification, because no entry method (usually Start) was present in the application. The application can be compiled and linked if desired, but any driver-specific link step should be suppressed.
Error Codes	Further steps that are dependent on a successful Batch Build should not execute. **8**-Generation failed. **12**-The application could not be opened. Generation was not attempted. This is usually due to an invalid value for either the APPHLQ= or APP= parameter. **16**-Input parameters are invalid. No action was taken.

A short execution log is written to STDOUT (usually SYSPRINT DD). It can generally be ignored when the return code indicates a success or a warning. When the return code indicates an error, the log should be consulted, as it will typically contain a descriptive message that can aid in resolving the problem.

More information:

Batch Build Procedures (see page 82)

Build an XPLINK Application

XPLINK (eXtra Performance LINKage) is an alternative module linkage convention that results in smaller, faster DLLs. AION is XPLINK-enabled, supporting applications built with XPLINK or traditional z/OS linkage. To enhance the performance of XPLINK-enabled user applications, Aion provides an XPLINK version of many of its own DLLs. These DLLs are automatically utilized by user applications built with the XPLINK option.

There are several XPLINK procedures to help users build XPLINK applications. To differentiate the XPLINK procedures from the non-XPLINK procedures, the names of the XPLINK procedures end with the letter X. The XPLINK procedures can be used in JCL constructed and submitted manually on the mainframe or automatically by the Remote Build option of the Windows development environment.

Note: When an application is built with the XPLINK option, the generated C/C++ programs must be compiled using the z/OS C/C++ compiler (CCNDRVR) with the XPLINK parameter, and XPLINK-compiled objects must be link-edited using the DFSMS Program Management Binder. The provided XPLINK procedures meet these requirements and must not be altered.

Build an XPLINK Application Using Batch Build

To build an XPLINK application using Batch Build

1. Use the XPLINK version of the build, compile, and link procedures.
2. Specify the argument XPLINK in the SYSIN file override to the BABUILDX procedure as shown in the following example:

```
//JOBCARD
//S01BLD EXEC BABUILDX,AIONHLQ=AION.V11R0M0
//PS1GEN.SYSIN       DD *
APP=USERAPP
APPHLQ=USERHLQ
LIB=AION.V11R0M0
WRAPPER=DRIVER
XPLINK
/*
```

The parameters for the remainder of the XPLINK procedures are exactly the same as for the non-XPLINK procedures. However, it is critical that only the XPLINK version of the procedures be used when compiling and linking an XPLINK application.

Sample JCL for compiling and linking a batch XPLINK application follows:

```
//S02COMPX EXEC BACOMPX,
//              APP=USERAPP
//              APPHLQ=USERHLQ
//              AIONHLQ=AION.V11R0M0
//*
//S03LINKX EXEC BALINKX,
//              APP=USERAPP
//              APPHLQ=USERHLQ
//              AIONHLQ=AION.V11R0M0
//              DLLLIB=USERHLQ.AION.LOAD
//*
//S04LNKDX EXEC BALNKDRX,
//              APP=USERAPP
//              APPHLQ=USERHLQ
//              AIONHLQ=AION.V11R0M0
//              DLLLIB=USERHLQ.AION.LOAD
```

Build an XPLINK Application Using Remote Development

To build an XPLINK application using remote build, specify the XPLINK parameter in the *Batch Build Parameters* window of the *Job Control* tab section in the *Remote Development Settings* area. This is the only additional step required to build an application with XPLINK. All proper XPLINK procedures required to build the application for XPLINK will be selected automatically.

Included Libraries in an XPLINK Application

The XPLINK version of any included system libraries are automatically utilized at runtime. However, any included *user* libraries must be built for XPLINK or the runtime environment will contain a mixture of XPLINK and non-XPLINK DLLs, reducing runtime performance due to the overhead of switching back and forth between linkage conventions.

Note: The XPLINK version of system libraries has a letter X in the second position of the name (rather than a letter E). The Aion runtime libraries depend upon this convention in order to automatically provide the correct system DLL to both XPLINK and non-XPLINK applications. Therefore, if a user wants to modify an AION system library for use with XPLINK applications, it must be built with XPLINK and the DLL name must have a letter X in the second position in its name. There are no requirements or restrictions on the composition of user DLL names (XPLINK or non-XPLINK) except that they must differ from the Aion system DLL names.

Edit Objects and Libraries in an Existing Application

When using Remote Development to build an application, it is important to know that it does not recursively build included libraries as the local build does. Therefore, you must build each included *user* library (not supplied libraries, such as SYSLIB or DATALIB) and copy the generated DLL files to the appropriate load libraries.

Execute Aion BRE Applications

Aion BRE applications can be stand-alone (load module plus DLL), or components (DLL only). To facilitate testing and debugging of logic, you can run stand-alone Aion applications in *interpreted* mode, that is, without building them.

Aion components can be built and then called by other programs. Calling programs can be C, C++, COBOL, or PL/I clients in CICS/IMS/TSO/batch, or they can be PC clients written in Aion (or other COM-supporting languages). Aion applications can also be accessed as MAES components from C/C++ or Java programs via TCP/IP.

To execute Aion BRE applications, use one of the following methods:

- Interpretively use the Remote Development feature on the mainframe directly from the Windows-based IDE. The result of this approach is that a batch job is executed.
- Use your own JCL on Windows (built using the Remote Development feature)-This results in a batch job execution.
- In TSO as a batch job (compiled).

As a component in MAES (compiled).

For more information see the *CA Aion BRE Product Guide*

More Information:

Interpreted Execution (see page 92)

Execute an Application in Batch (see page 92)

Run an Application as a MAES Component (see page 96)

Interpreted Execution

You can execute your application interpretively using one of the following methods:

- From Batch-Use the provided JCL procedure BARUN to invoke REEXEC*nn* in the Aion Load library. Alternately, the procedure BARUNX can be used to invoke RXEXEC*nn,* the XPLINK version of REEXEC*nn*.

  Note: For r11.0 the module name is REEXECB0, or RXEXECB0.

- From the Windows-based IDE-Use the Remote Development feature of the Windows version of CA Aion BRE to generate JCL to execute an application in batch using program REEXEC*nn or RXEXECnn*.

  Note: For more information, see the *CA Aion BRE Product Guide.*

More Information:

Execute an Application in Batch (see page 92)

Execute an Application in Batch

You can execute an Aion application (compiled or interpreted) as a stand-alone program directly from your own JCL, using a distributed JCL procedure, or via a CLIST. To execute from a JCL procedure, use BARUN or BARUNX in the procedure library distributed with the Aion product. Alternately, you can invoke REEXEC*nn* or RXEXEC*nn* directly from your own JCL.

Note: For r11.0 the module name is REEXECB0, or RXEXECB0.

To execute an application in Batch

1. Copy the HELLO OR HELLOX execution JCL member provided with Aion. This JCL is unloaded when Aion is installed, and can typically be found in <aionhlq>IVP.JCL.

   aionhlq represents the unique high-level qualifier that was originally specified for the Aion installation.

2. Modify the execution JCL to specify the following:

   - Your job card information
   - The high-level qualifier for your Aion installation
   - The load library containing built Aion applications and compiled Aion included libraries

3. Use the execution JCL for both interpreted and compiled execution.

   Note: REEXEC*nn* and RXEXEC*nn* can be used interchangeably for executing applications interpretively as long as all the included user libraries are being executed interpretively. However, care must be taken if any of the included libraries are to be executed compiled, because all compiled libraries must be compiled the same way (either with or without the XPLINK option). In addition, if the included user libraries are compiled with the XPLINK option, RXEXEC*nn* must be used. Conversely, if the included user libraries are compiled without the XPLINK option, REEXEC*nn* must be used.

For Interpreted Execution

To run an application interpretively, execute REEXECnn or RXEXECnn (see previous note), specifying as a parameter (PARM=) the Aion library high-level qualifier, and the name of the application you want to run. The Aion-provided JCL procedures BARUN and BARUNX execute REEXEC*nn* and RXEXEC*nn*, respectively.

Note: For r11.0 the module name is REEXECB0, or RXEXECB0.

To illustrate, the following shows execution JCL that is customized to run an application interpretively. The application name is MYAPP. Statements preceded by //* will be ignored during execution.

```
//AIONEXEC JOB (12345),'EXECUTE AION KB',MSGCLASS=X

//*
//APPEXEC EXEC PGM=REEXECnn,PARM='aionhlq MYAPP'
//*
//STEPLIB DD DSN=aionhlq.LOADLIB.LOAD,DISP=SHR
//                DD DSN=aionhlq.LOAD,DISP=SHR
//*
//* NOTE: MESSAGEBOX OUTPUT SENT TO SYSPRINT
//SYSPRINT DD SYSOUT=*
//SYSOUT  DD SYSOUT=*
//AIONTRAC DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
```

Note: An Execution Trace file is always produced in interpreted mode. If you have identified an Execution Trace data set in the JCL (using the AIONTRAC DD), Execution Trace output is written to that data set. If you do not explicitly identify a data set, interpreted execution output is written to a data set named *userid.<aionhlq>.appname.*TRACE.

You can selectively control execution trace output by using method TraceEnable in _utilities class. For example:

```
Traceenable( false ) // deactivates tracing
Traceenable( true ) // reactivates tracing
```

This allows you to reduce the amount of trace output that is produced.

For Compiled Execution

For compiled execution, execute the load module for the built Aion application. In our sample customized execution JCL, $MYAPP is the name of the compiled application.

The following sample JCL is the same as that used previously to illustrate interpreted execution. The REEXEC*nn* statement is now preceded with //* and will be ignored. The execution statement that follows it (//BLTEXEC) has been modified to remove the asterisk so it will be processed.

```
//AIONEXEC JOB (12345),'EXECUTE AION KB',MSGCLASS=X
//*
//BLTEXEC EXEC PGM=$MYAPP
//*
/STEPLIB DD DSN=userhlq.APP.LOADLIB,DISP=SHR
//       DD DSN=aionhlq.LOAD,DISP=SHR
//*
//* NOTE: MESSAGEBOX OUTPUT SENT TO SYSPRINT
//SYSPRINT DD SYSOUT=*
//SYSOUT  DD SYSOUT=*
//AIONTRAC DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CEEDUMP DD SYSOUT=*
```

Submit the customized execution JCL to run the application.

When execution is initiated via the generated driver program ($MYAPP), execution of Aion XPLINK applications is identical to that of Aion non-XPLINK applications. However, if an Aion XPLINK DLL is accessed via a COBOL, PL/1, or C client program, the client program must be initiated with the LE parm XPLINK(ON) to activate XPLINK support.

Note: The $MYAPP module is located via a STEPLIB search. In this example, it would either be in the user's library userhlq.APP.LOADLIB, or in the Aion library aionhlq.LOAD.

While it is not recommended (due to performance considerations), you can execute an Aion application containing a mixture of XPLINK and non-XPLINK DLLs. For mixed applications, as long as the main application is compiled with the XPLINK option, no JCL changes are necessary. However, if the main application is not compiled with the XPLINK option, you must invoke the application with the LE parameter XPLINK(ON). You can also specify the HEAPP(ON) parameter to improve performance, but it is not required. Example JCL specifying LE runtime options follows:

```
//BLTEXEC EXEC PGM=$MYAPP,PARM='XPLINK(ON),HEAPP(ON)/'
```

Run an Application as a MAES Component

You can run an Aion application as a server component in MAES. The client can drive from a PC, from a supported mainframe TP monitor (CICS or IMS), or via TCP/IP.

To run an Aion application as a server component in MAES

- Build the Aion server application on z/OS with the appropriate build directives.
- Route the generated Aion DLL to a load library available to MAES.
- For XPLINK components, XPLINK(ON) must be specified in the MAESAPRM dataset.
- Give TP monitor clients or the PC clients the node name of MAES.
- Give TCP/IP clients the host name and port number of MAES.
- If using IMS, include a local LU name.

More Information:

XPLINK MAES Components (see page 180)

PC Clients

PC clients access a proxy that is generated on the PC from the server application. If using a PC client, build the server component on the PC using the Z/OS COM Interface Layer to generate the proxy.

More Information:

Create and Use Aion Components for CICS Clients (see page 116)

TP Monitor Clients

A TP monitor client is written in C, PL/I, or COBOL to interact with CICS or IMS data using a built Aion component.

More Information:

Build and Manage Aion Components (see page 107)

TCP/IP Clients

TCP/IP clients written in C, C++, or Java can interact with a built Aion component that is defined in a MAES environment. Refer to the CLIMSTP example which is a C++ program that interacts via TCP/IP with the ARGSERV.APP running in MAES. Details regarding the CLIMSTP example are described in <aionhlq>.SAMPLES.README(TCPIP).

More Information:

Build and Manage Aion Components (see page 107)

COBOL and PL/I User Applications

User programs written in COBOL or PL/I should include one of these SDS files. There is a generic client DLL of the same name that loads in the CICS or IMS region:

Suspend Version	Non-Suspend Version
RECCOB*nn* CICS COBOL	RECCON*nn* CICS COBOL
REICOB*nn* IMS COBOL	REICON*nn* IMS COBOL
RECPLI*nn* CICS PL/I	RECPLN*nn* CICS PL/I
REIPLI*nn* IMS PL/I	REIPLN*nn* IMS PL/I

Note: For r11.0 the member suffixes (nn) are B0.

When you build an Aion application with a COBOL or PL/I interface, CA Aion BRE generates a copybook of data definition statements to be copied into the data division of the user program. Each call occupies one *communications area*. An identical call is made for the method that is invoked, specifying the communications area for the desired method. This is where the parameter values have been placed, and from where returned parameter values can be fetched.

The COBOL stub calls the appropriate Aion wrapper:

Suspend/Resume Version	Non-Suspend Version
RECWS*nn*-CICS Wrapper	RECWN*nn*-CICS Wrapper
REIWS*nn*-IMS Wrapper	REIWN*nn*-IMS Wrapper

Note: For CICS, all programs must be defined to CICS, including the user program, the client application DLL (C clients only), the Aion stub, the wrappers, and RESYS*nn,* REUTIL*nn* and RECOMD*nn*. The SDS file that is specified when the user program is link-edited determines the wrapper type.

More Information:

Suspend/Resume MAES Components ('Hot' Apps) (see page 181)

Run Aion Components from Batch COBOL Programs

Important! Because Aion components are built as dynamic link libraries (DLL files), only COBOL programs compiled with r3.3 (or higher) of the Enterprise COBOL for z/OS compiler can access Aion components in batch.

An Aion component DLL exports a series of functions written in the C language. These functions are described in a side definition (SDS) generated when linking the component DLL. A COBOL program that calls these functions must be linked with this SDS (using either the LE prelinker or the z/OS binder), and the program must be compiled using the following compiler options:

```
'DLL,PGMNAME(LONGMIXED),NODYNAM,RENT'
```

The COBOL compiler options instruct the compiler to treat all calls as DLL function calls, and to support long function names. The component functions can then be used exactly as described in the application header file, saved at build time in *<aionhlq>.appname.bin(appname)*.

The following figure shows a sample header file:

```
000023 //* Class definitions for AddClass
000024 //* Methods
000025
000026 val_InPtr EXPORT _cdecl AddClass_Create (LPTSTR name);
000027
000028 VOID EXPORT cdecl AddClass_Delete (val_InPtr pin);
000029
000030 long EXPORT _cdecl AddClass_Add (val_InPtr pin,long int1,long int2);
```

All input, output, and return arguments must be represented using the corresponding COBOL data types. All simple data types are passed BY VALUE, and strings are passed BY REFERENCE. Output arguments are always passed BY REFERENCE.

Important! Strings that are passed as output arguments must be valid Aion string pointers passed BY REFERENCE.

To get a valid Aion string pointer, call the xs_Make( ) function in RESYS*nn*.

The following shows the xs Make( ) function:

```
000600            CALL 'xs_Make' USING
000600                BY REFERENCE NULLPOINTER
000600                RETURNING ASTRING-PARG1.
000631            CALL 'ostrings_mstring_o' USING
000632                BY VALUE OSTRINGS-HINST
000633                BY REFERENCE ASTRING-PARG1
000634                BY VALUE 1.
000635
000636            SET ADDRESS OF LNK-STRING1 TO ASTRING-PARG1.
```

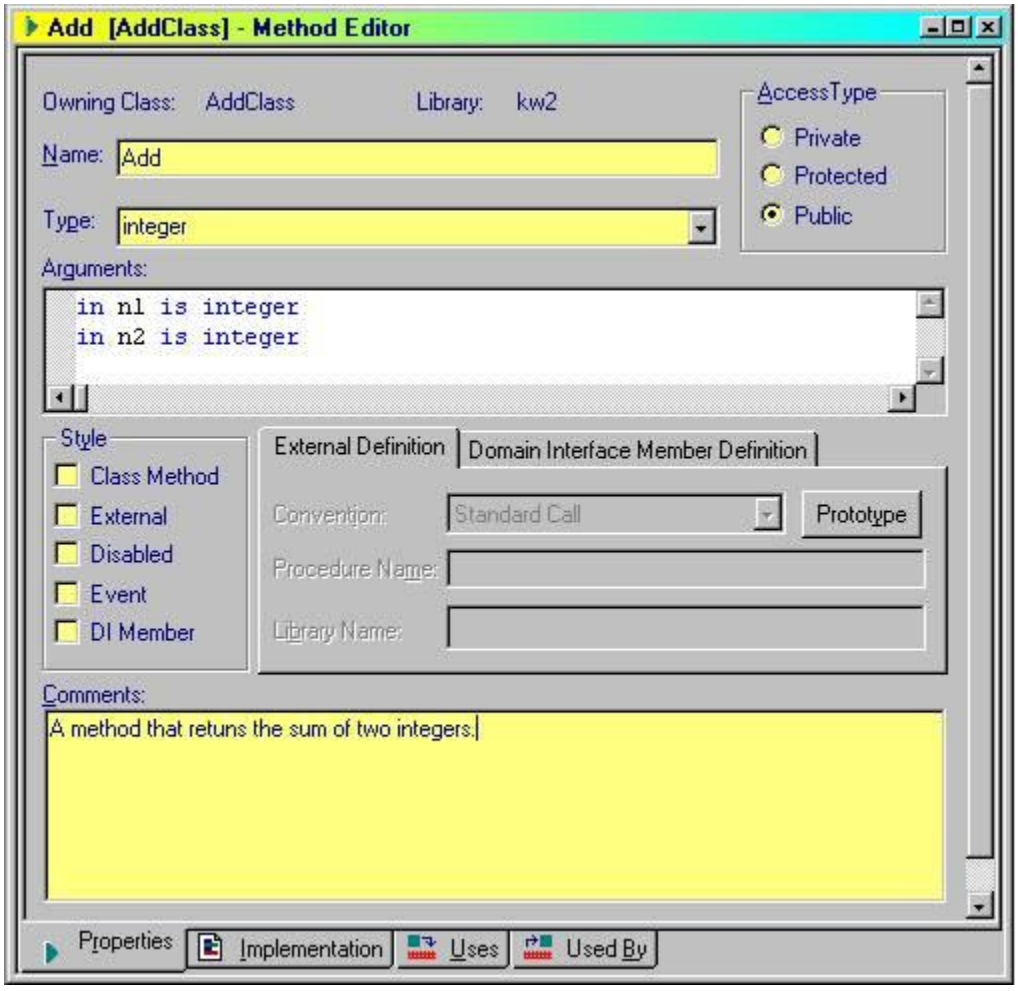
The first argument for each instance-method must be the instance handle returned by the class_Create function.

PreLinking

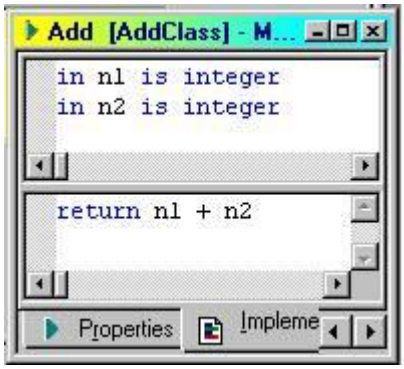
When linking the COBOL program, include the generated SDS file *<aionhlq>.appname.build(SDS)* into the pre-link step (see following samples).

Important! At execution time, the load library containing the Aion runtime DLL files must be in the STEPLIB concatenation.

This is an example of a COBOL program calling a method named Add in a class named AddClass within an Aion application ADDER.APP. The Add method is defined in the Aion IDE as follows:



This Add method has two integer input parameters.



The ADDER.APP Aion application is built with the WRAPPER=COBOL directive, before the COBOL program is compiled. The SDS(SDS) member that is created when the ADDER.APP program is built is referenced by the COBOL program's linker input instructions.

The following figure shows sample linker input:

```
000100   INCLUDE REOBJ(BCOBADD)
000110   INCLUDE SDS(SDS)
000200   NAME BCOBADD(R)
```

The following figure shows a sample COBOL Compile and Link PROC:

```
//COB390  PROC LNGPRFX='IGY.V2R1M0',
//*            SYSLBLK=3200,
//             LIBPRFX='CEE',PLANG=EDCPMSGE,
//             PGMLIB='&&GOSET',GOPGM=GO
//*******************************************************************
//* IBM COBOL FOR z/OS & VM VERSION 2 RELEASE 1 MODIFICATION 0
//* LICENSED MATERIALS - PROPERTY OF IBM
//*******************************************************************
//*  COMPILE, PRELINK AND LINK EDIT A COBOL PROGRAM
//*
//*  PARAMETER  DEFAULT VALUE    USAGE
//*   LNGPRFX   IGY.V2R1M0       PREFIX FOR LANGUAGE DATA SET NAME
//*   SYSLBLK   3200             BLOCKSIZE FOR OBJECT DATA SET
//*   LIBPRFX   CEE              PREFIX FOR LIBRARY DATA SET NAMES
//*   PLANG     EDCPMSGE         PRELINKER MESSAGES MODULE
//*   PGMLIB    &&GOSET          DATA SET NAME FOR LOAD MODULE
//*   GOPGM     GO               MEMBER NAME FOR LOAD MODULE
//*
//*  CALLER MUST SUPPLY //COBOL.SYSIN DD ...
//*
//COBOL  EXEC PGM=IGYCRCTL,REGION=2048K,
//             PARM='DLL,RENT,PGMNAME(LONGMIXED)'
//STEPLIB  DD  DSNAME=&LNGPRFX..SIGYCOMP,
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DISP=SHR,DSN=PDJXL.COBOL.OBJ(&GOPGM)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//*******************************************************************
//PLKED   EXEC PGM=EDCPRLK,PARM='',COND=(8,LT,COBOL),
//             REGION=2048K
//STEPLIB  DD  DSNAME=&LIBPRFX..SCEERUN,
//             DISP=SHR
//SYSMSGS  DD  DSNAME=&LIBPRFX..SCEEMSGP(&PLANG),
//             DISP=SHR
//SYSLIB   DD  DUMMY
//REOBJ DD DISP=SHR,DSN=<userhlq>.COBOL.OBJ
//SDS DD DISP=SHR,DSN=<userhlq>.BUILD
//SYSIN DD DSN=<userhlq>.COBOL.CTL(&GOPGM),DISP=SHR
```

```
//SYSMOD   DD  DSNAME=&&PLKSET,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(32000,(100,50)),
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSDEFSD DD  DUMMY
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//*******************************************************************
//LKED   EXEC PGM=HEWL,COND=((8,LT,COBOL),(4,LT,PLKED)),REGION=1024K
//SYSLIB   DD  DSNAME=&LIBPRFX..SCEELKED,
//             DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIN   DD  DSNAME=&&PLKSET,DISP=(OLD,DELETE)
//         DD  DDNAME=SYSIN
//SYSLMOD  DD  DSNAME=&PGMLIB(&GOPGM),
//             SPACE=(TRK,(10,10,1)),
//             UNIT=SYSDA,DISP=(MOD,PASS)
//SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(10,10))
```

More Information:

Fine Tune Application Performance (see page 104)

Run Aion XPLINK Components from Batch COBOL Programs

For a batch COBOL program to call XPLINK DLLs, you must specify the LE option XPLINK(ON). For better performance, also specify the LE option HEAPP(ON). For optimal performance, additional LE options relating to stack and heap allocation may be required.

Note: that the LE option CBLOPTS determines whether the LE options precede or follow the slash ("/") in the parameter value specified for COBOL main programs. For example, if your LE environment has been set up with CBLOPTS(ON), then application parameters precede runtime parameters and your execution JCL should look like the following:

```
//STEP1   EXEC PGM=COBOLPGM,PARM='/XPLINK(ON),HEAPP(ON)'
```

However, if your LE environment has been set up with CBLOPTS(OFF), runtime parameters precede application parameters and your execution JCL should look like the following:

```
//STEP1   EXEC PGM=COBOLPGM,PARM='XPLINK(ON),HEAPP(ON)/'
```

Contact your systems programmer to determine how to properly construct the parameter value at your installation.

Execute External Programs from an Aion Component

CA Aion BRE makes it easy to call C/C++ and COBOL DLLs, and z/OS assembler programs from a mainframe component.

- To call C/C++ and COBOL DLLS from CA Aion BRE:

  Call a C/C++ or COBOL program by using the external method facility in CA Aion BRE. Calling a COBOL DLL by this method is straightforward: the Aion string, integer, and real datatypes can be directly sent to and received from a COBOL program.

- To call Assembler programs from CA Aion BRE:

  Use the RunMVSProgram() method to call a z/OS assembler program. (The RunMVSProgram method is provided by the _Utilities class in Syslib.)

Note: Whenever an XPLINK DLL (Aion or C/C++) is called from an Aion component, the main program must be linked XPLINK or invoked with XPLINK(ON) or a runtime error condition will result. The step that is necessary depends upon the execution environment (batch or online).

For batch execution, if the Aion application is built with the XPLINK option and WRAPPER=DRIVER, the generated driver (main) program is automatically linked XPLINK.

For online execution (Aion components executing in MAES), XPLINK(ON) must be specified in the MAESAPRM dataset. For more information refer to the description of the MAESAPRM DD in the "Specify MAES DD Statements" section.

For more information, see the "Generate and Use C and C++ Components" in the *CA Aion BRE Product Guide*

More Information:

XPLINK MAES Components (see page 180)

Fine Tune Application Performance

Because Aion applications and Aion runtime libraries execute as z/OS C/C++ programs, they execute under the control of the IBM LE runtime libraries. When executing an Aion application, you can specify IBM LE runtime options to monitor and tune memory utilization. Tuning memory use of Aion applications can have a great impact on application CPU time performance and memory use.

Typically, the RPTSTG (Storage Report), HEAP, STACK, and HEAPPOOLS options are used in the following scenario.

To fine tune application performance

1. Execute your Aion application and record the used CPU time (in the JES output).
2. Execute with the Storage Report ON to see what recommendations the C runtimes provide for your application.
3. Specify the suggested options and re-run your application to see if the CPU time is decreased.

Some LE runtime options (such as HEAPPOOLS) may require several iterations to determine the best values for your application. These options and the storage report are described in detail in IBM's z/OS Language Environment Programming Reference. For more information about using LE memory parameters to achieve optimal performance, see IBM's z/OS Language Environment Customization, sections Tuning Stack Storage and Tuning Heap Storage.

Aion applications tend to be very logic-intensive and CPU-intensive. Applications that create and delete many instances can also be very memory intensive, as can allocation and freeing of memory. The LE runtime report can tell you how much memory (HEAP) your application uses. In some cases, simply preallocating the recommended initial heap size (using the HEAP runtime option) has been known to improve memory-intensive processing (such as creating and assigning instance values) by up to 10 times over processing times using the default HEAP size.

For batch Aion applications, the LE parms can be specified in the execution JCL's PARM field (for example, EXEC PGM=BUILTAPP,PARM=('LE parms/')). For Aion applications executing in MAES, there are two ways LE parms can be specified.

- One way is for a CEEUOPT module specifying the desired LE parameters to be linked with the OAES module in <aionhlq>.MAES.LOAD. The <aionhlq>.MAES.JCL member CEEUOPT contains a sample job for use in evaluating the STACK and HEAP options for Aion applications running in MAES. Parameters specified in a CEEUOPT file apply to all applications executing in MAES.
- A second way (only available for applications accessed via the ACQUIRE function) is to specify the desired LE parameters in the MAESAPRM file. Any values specified via the MAESAPRM file will override values specified in the CEEUOPT file.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

Specify C or LE Runtime Options

With z/OS, the C runtime options are documented along with the LE parms in IBM's z/OS *V1R2.0 Language Environment Programming Reference*. Specify C or LE runtime options for Aion applications in the ways described in the following paragraphs.

For Stand-alone Applications

For stand-alone Aion applications, the runtime options can be specified as parameters in the JCL used to execute the application. The arguments are separated by a comma and must end with a forward slash (/). For example:

```
//STEP01 EXEC PGM=$LOANS,PARM='ALL31,TRACE/'
```

For MAES or Batch Components

Aion applications executing in MAES are DLLs. As a result, you cannot specify their runtime options from the calling client program. There are two ways to specify runtime values for Aion applications executing in MAES:

- Specify the runtime values for all components in the MAES region by creating a CEEUOPT module and linking CEEUOPT into the OAES module in <aionhlq>.MAES.LOAD. The <aionhlq>.MAES.JCL member CEEUOPT contains a sample job for use in setting runtime options for CA Aion applications running in MAES.
- Specify runtime values individually for CA Aion components invoked using the Acquire function. The desired values are defined in the MAESAPRM file. Any values specified via the MAESAPRM file will override values specified in the CEEUOPT file.

More Information:

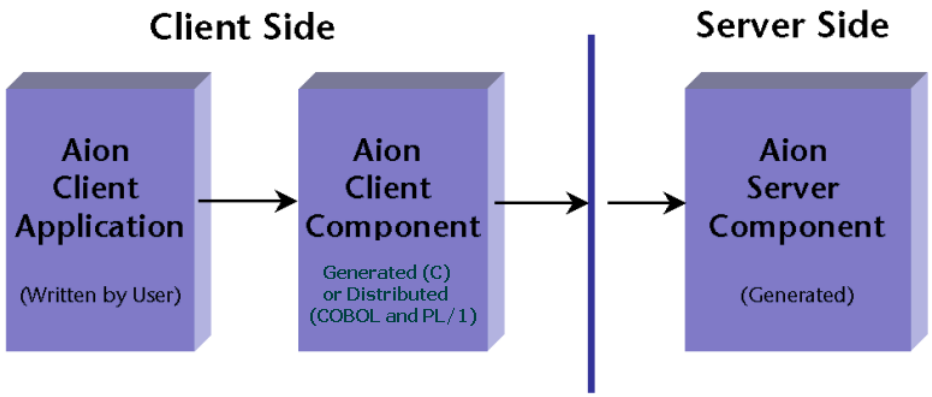
Specify MAES Runtime Parameters and DD Statements (see page 192)

Chapter 4: Build and Manage Aion Components

CA Aion BRE client-server architecture is based on a model in which each method or function call (argument, parameter, return code) is shipped from the client to the server. For each application, the client and server components are generated separately. The server component is an Aion application hosted by MAES and generated using the C language interface.

This section contains the following topics:

This figure shows, the Aion client application calls functions in the Aion client DLL. The Aion client DLL forwards the function calls to MAES, where they are executed by the appropriate Aion server DLL.



Generate an Aion Component

To generate an Aion component

1. Edit the Aion server application.
2. Verify the public status of instance methods you want exposed to the client. These public methods must reside in a class that has its export property checked.
3. Specify the appropriate TP interface (CICS or IMS) and client language (COBOL, PL/I, or C) in the build directives.
4. Build the application.

For specific information about generating clients, see the following topics:

- Create and Use Aion Components for CICS Clients-You can generate an interface component to enable communication between an Aion server and conversational or pseudo-conversational CICS transaction client programs written in COBOL, PL/I, or C.
- Create and Use Aion Components for IMS Clients-You can generate an interface component to enable communication between an Aion server and conversational or non-conversational IMS transaction client programs written in COBOL, PL/I, or C.
- Create and Use Aion Components for use in MAES by a PC Client-CA Aion BRE supports PC clients using APPC or TCP/IP to access mainframe server components.

More Information:

Create and Use Aion Components for CICS Clients (see page 116)
Create and Use Aion Components for IMS Clients (see page 130)
Create and Use Aion Components on PC Clients (see page 160)

MAES Clients

With a MAES client, the client application and the Aion server DLL execute in different address spaces (potentially on different machines). When you build an Aion component for a remote client, CA Aion BRE generates two pieces:

- The server component DLL that runs on the mainframe (in MAES).
- The client component, which is generated on the mainframe for CICS and IMS clients or on the PC for PC clients.

IMS and CICS Clients

IMS and CICS Clients consist of the following procedures:

- Building a COBOL Client
- Building a PL/I Client
- Building a C Client

Build a COBOL Client

Building an IMS or CICS COBOL client generates the server component DLL. It also generates a COBOL copybook that contains data definitions for all exported methods in the Aion component. This copybook can be copied to the SYSLIB concatenation of the COBOL compiler, and can then be included by the client COBOL program using the COBOL COPY command. Aion also generates a sample library member that contains sample COBOL code for calling each method within the Aion component.

Build a PL/I Client

Building an IMS or CICS PL/I client generates the server component DLL. It also generates a PL/I include member that contains data definitions for all exported methods in the Aion component. This include member can be copied to the SYSLIB concatenation of the PL/I compiler, and then be included into the PL/I client program using the %INCLUDE PL/I command. Aion also generates a sample library member, which contains sample PL/I code for calling each method within the Aion component.

Building a C Client

Building an IMS or CICS C client generates a server component DLL and a client component DLL, exporting all public functions in the component and exposing these functions and their parameters to the client C program. A header file is also generated, containing C function prototypes calling each method within the Aion component.

C Clients

For IMS or CICS C clients, the client component is an application-specific client DLL that includes the same exported methods as the actual component. The client DLL's only job is to ship the client's function calls to and from the server component DLL using a MAES-specific message protocol.

In order to resolve the Aion function calls, the link-edit for user programs written in C must include the SDS file for the generated client DLL. The SDS file is located in *<aionhlq>.appname.*BUILD2*(*SDS*).* The generated client DLL must be available to be loaded in the associated CICS or IMS region at run time.

By default, the generated client DLL calls the Suspend/Resume wrapper (communication DLL). The CONVERSATIONAL parameter can be specified when building the Aion application to specify the Non-Suspend wrapper. The Non-Suspend wrapper is required for client programs that issue SUSPEND, RESUME, ACQUIRE, or RELEASE functions.

The SUSPEND and RESUME functions are used by client applications that access client-managed, reusable apps ("hot apps") executing in MAES. See the sample client programs ending in '2' for examples: <userhlq>.SAMPLES.SOURCE(CLCICCB2).

The ACQUIRE and RELEASE functions are used by client applications that access MAES-managed, reusable apps ("hot apps") executing in MAES. See the sample client programs ending in '3' for examples: <userhlq>.SAMPLES.SOURCE(CLCICCB3). The ACQUIRE and RELEASE functions are also used to access MAES systems participating in a VTAM Generic Resource pool. See the sample client programs ending in '3' for examples.

Suspend/Resume Version	Non-Suspend Version
RECWS*nn*-CICS Wrapper	RECWN*nn*-CICS Wrapper
REIWS*nn*-IMS Wrapper	REIWN*nn*-IMS Wrapper

Note: For r11.0 the module suffix (nn) is B0.

Note: For CICS, all programs must be defined to CICS, including the user program, the generated or generic client DLLs, the communication DLLs, and the RESYS*nn*, REUTIL*nn*, and RECOMD*nn* DLLs.

The Suspend/Resume and Acquire/Release modes of operation are discussed in Suspend/Resume MAES Components ("Hot" Apps).

COBOL and PL/I Clients

Important! Because online COBOL and PL/I clients access Aion component DLLs by calling generic client DLLs, both CICS and IMS client programs must be compiled with DLL linkage.DLL linkage is also required to utilize the "helper functions" demonstrated in the sample client programs supplied with CA Aion BRE in <aionhlq>.SAMPLES.SOURCE and described in the Sample Client Programs section.

COBOL clients must be compiled with r3.3 (or higher) of the Enterprise COBOL for z/OS compiler using options 'DLL,PGMNAME(LONGMIXED),NODYNAM,RENT'. These options instruct the COBOL compiler to treat all calls as DLL function calls and to support long function names. The component functions can then be called as described in the sample code file saved at build time in *<aionhlq>.appname.COBOL(appname)*.

PL/I clients must be compiled with r3.3 (or higher) of the Enterprise PL/1 for z/OS compiler. The PL/1 compiler does not require special options to treat calls as DLL function calls and to support long function names. The component functions can be called as described in the sample code file saved at build time in *<aionhlq>.appname.PL1(appname)*.

For IMS or CICS COBOL and PL/I clients, the client incorporates the code fragments generated in the *<aionhlq>.appname.COBOL(appname)* or *<aionhlq>.appname.PL1(appname)* file to call a generic client DLL. The generic client DLL's only job is to ship the client's function calls to and from the server component DLL using a MAES-specific message protocol.

In order to resolve the Aion function calls, the link-edit for user programs written in COBOL or PL/I must include an SDS file for one of the generic client DLLs. The generic client DLL of the same name must be available to be loaded in the associated CICS or IMS region at run time. The nn suffixes below are associated with the Aion BRE version you are using. For r11.0 the nn suffix is B0.

Suspend Version	Non-Suspend Version
RECCOB*nn* CICS COBOL	RECCON*nn* CICS COBOL
REICOB*nn* IMS COBOL	REICON*nn* IMS COBOL
RECPLI*nn* CICS PL/I	RECPLN*nn* CICS PL/I
REIPLI*nn* IMS PL/I	REIPLN*nn* IMS PL/I

When you build an Aion application with a COBOL or PL/I interface, CA Aion BRE generates a copybook of data definition statements to be copied into the data division of the user program. Each exported public method in the Aion app has an associated *communications area*. The call to the generic client DLL specifies the communications area for the desired method. This communications area is where the input parameter values are placed before the call and where the output and/or returned parameter values are retrieved after the call.

The SDS file that is specified when the user program is link-edited determines the wrapper type because each client DLL calls the corresponding Aion wrapper (communication DLL). The nn suffixes below are associated with the Aion BRE version you are using. For r11.0 the nn suffix is B0.

Suspend/Resume Version	Non-Suspend Version
RECWS*nn*-CICS Wrapper	RECWN*nn*-CICS Wrapper
REIWS*nn*-IMS Wrapper	REIWN*nn*-IMS Wrapper

Note: For CICS, all programs must be defined to CICS, including the user program, the generated or generic client DLLs, the communication DLLs, and the RESYS*nn* and REUTIL*nn*, and RECOMD*nn* DLLs.

The Suspend/Resume mode is discussed in Suspend/Resume MAES Components.

More Information:

Supported Data Types in Arguments (see page 115)

Sample Client Programs

CA Aion BRE provides sample IMS and CICS client programs. Please refer to <userhlq>.SAMPLES.README(CLIENT) for specific examples. Their member names are coded according to the following legend:

Position	Codes
Program type	CL (Client program demonstrating object attribute exchange) SO (Client program demonstrating serialized object exchange)
Environment	CIC (for CICS) IMS BAT (for Batch)

Position	Codes
Language	CB (for COBOL) PL (for PL/I) C and C++ TP (for TCP/IP - currently for C/C++ language)
Other	2 - Manual suspend/resume 3 - Acquire/Release

For example, the sample client program CLIMSCB illustrates the use of COBOL for the IMS environment.

Aion Components

MAES lets you run Aion components as subtasks within the MAES address space. CA Aion BRE uses client interface components on CICS, IMS, or Windows to invoke components.

Client Component Interfaces

The language interface that is used when generating the client component DLL depends on the type of client and the client programming language being used.

- Windows clients use the Microsoft COM interface type.
- The mainframe supports COBOL, PL/I, and C language interfaces.

The COBOL or PL/I language types generate sample code to describe all functions, parameter lists, and return values, as well as sample code that calls the Aion COBOL or PL/I client DLL (a generic DLL distributed with the CA Aion BRE product).

The C language type generates a client component DLL that exports all functions contained in the server component DLL.

Aion Strings in COBOL and PL/I

Exported Aion methods that have string output (return values or output arguments) use the internal Aion string format. The Aion string must be converted to a format recognizable by the COBOL or PL/I program. CA Aion BRE provides the string manipulation functions required to convert an Aion string to and from a COBOL or PL/I string as well as other useful string manipulation functions. See the sample programs for details. Contact your systems programmer to locate the Aion sample programs. For more information refer to <userhlq>.SAMPLES.SOURCE for specific COBOL and PL/I examples.

Aion Strings in C

The functions that are available for converting C strings to and from Aion strings are documented in the section Handling Aion Strings in C and C++ in the chapter "Generate and Use C and C++ Components" in the *CA Aion BRE Product Guide*. The proper use of the string manipulation functions is demonstrated in the sample programs. Contact your systems programmer to locate the Aion sample programs. For more information refer to <userhlq>.SAMPLES.SOURCE for specific C examples.

Client Program Error Handling in MAES

Errors generate return code values for MAES clients executing in CICS or IMS. You can add logic to your client program to perform error handling according to the error code returned.

For COBOL, each 02 *methodname*-METHOD for CICS and IMS contains two fields; *methodname*-RC and *methodname*-REASON. For PL/I clients, the return code and reason fields are named OBJECT_*methodname*_RC, and OBJECT_*methodname*_REASON, respectively.

The status of each call to MAES is reported in these two fields; both are returned with a value of zero when the call was successful.

Supported Data Types in Arguments

The following data types are supported for input, output, and return arguments for the supported client application languages (COBOL, PL/I, and C/C++) executing in both the batch and online environments (CICS and IMS):

- BINARY STRING
- INTEGER
- ISOTIME
- DATE
- TIME
- TIMESPAN
- TIMEVAL
- BOOLEAN
- REAL
- STRING

In addition, the BINARY data type is supported for input and output arguments for COBOL, PL/I, and C/C++ clients operating in a batch environment, and for COBOL clients operating in the online environments (CICS and IMS).

Aion applications operating in a MAES TCP/IP environment can also be accessed by remote Java clients. Additional data types are available for these intersystem calls.

More Information:

Create and Use Aion Components for TCP/IP Clients (see page 142)

Access CICS Programs from MAES

The return value from RunRemoteProgram contains the VTAM sense codes described in the IBM Communications Server Bookshelf in book *CS: IP and SNA Codes*. The return value should be translated from decimal to hexadecimal to obtain an 8-character hexadecimal number composed of two 4-character hexidecimal numbers. The first number is the sense code and the second number contains either zero (0) or sense code-specific information (a sub-code).

Create and Use Aion Components for CICS Clients

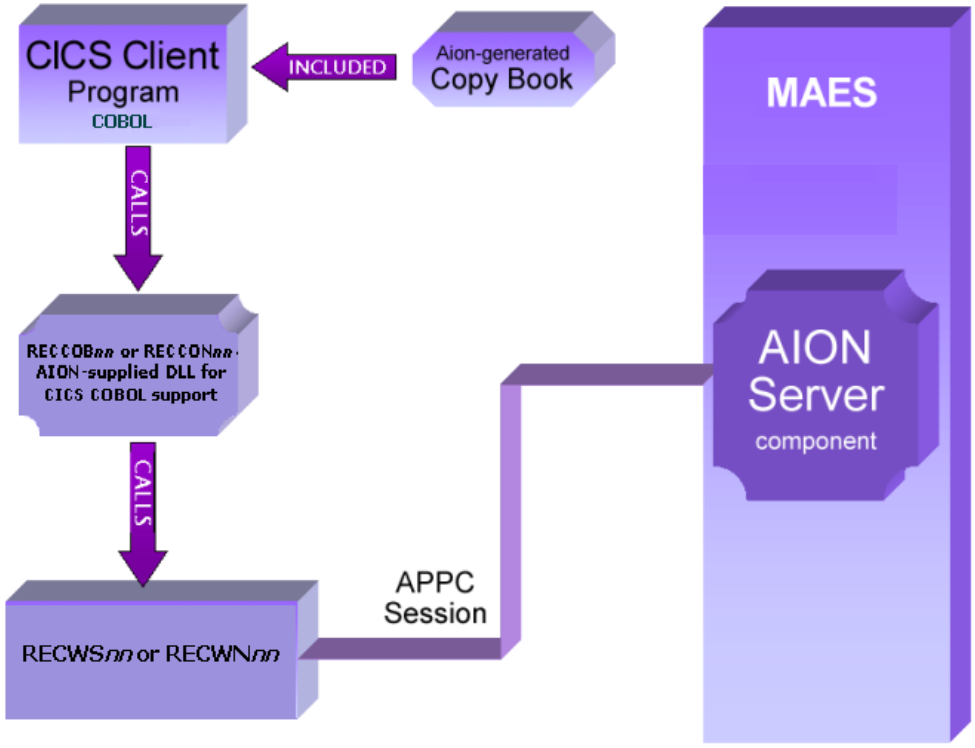
You can generate an Aion interface component to enable communication between an Aion server component and conversational or pseudo-conversational CICS transaction programs (client programs) written in COBOL, PL/I, or C.

- To generate a CICS COBOL-compatible or CICS PL/I-compatible Aion component, build your Aion server application with a COBOL-CICS or PLI-CICS interface layer, respectively. This generates the server component itself. It also generates a copybook (file of data definition statements) to be copied into the data definition area of the COBOL or PL/I user program.
- To generate a CICS C-compatible Aion component, build your Aion server application with a C-CICS interface. This generates the Aion server component DLL. It also generates a header file and a client component DLL that provides access to all exported public functions in the Aion server component DLL.

Generate a CICS COBOL Component

Use the COBOL-CICS interface layer to generate sample COBOL data definitions describing the parameters and return values for each individual function within the component. CA Aion BRE also generates sample COBOL code to call each of those functions.

The following illustrates a CICS COBOL client program communicating with an Aion server component in MAES.



In the illustration, RECCOB*nn* is the Suspend version. The Non-Suspend version is named RECCON*nn*. The Aion calls in the COBOL client program always specify RECCOB*nn*. The SDS file linked with the COBOL client determines whether RECCOB*nn* or RECCON*nn* actually is invoked.

Note: For r11.0 the module suffix (nn) is B0.

Specify the CICS COBOL Build Directives

There are two ways to specify build directives and generate the required client and server components:

- Use Batch Build with the parameter WRAPPER=COBOL-CICS. Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.
- Remote Build capabilities on the Windows workstation.

The following are produced when generating the CICS COBOL component:

Generated	Purpose	Location
Sample COBOL Code	Cut and paste into the COBOL program for use with the copybook.	*<aionhlq>.appname*.COBOL(SAMPLE)
Copybook	Include in the COBOL program (using the COPY statement). The copybook contains data definition statements comprising one communications area for each call.	*<aionhlq>.appname*.COBOL(*appname*)

Note: The generated COBOL external definition files include strings containing the names of exported classes and methods. The processing of these strings in MAES is case sensitive. When editing generated COBOL copybooks, do not modify the case of string values containing class and method names.

To generate field names, Aion uses the names of an application's class, method and argument, concatenated with other text.

For more information, see the *CA Aion BRE Product Guide*.

More Information:

Build and Manage Aion Applications (see page 71)

More Information:

Supported Data Types in Arguments (see page 115)

Use the CICS COBOL Component

The generated copybook contains a complete communications area for each method (INIT-METHOD, SUSPEND-METHOD, RESUME-METHOD, ACQUIRE-METHOD, RELEASE-METHOD, CLEANUP-METHOD, *objname*-CREATE-METHOD, *objname*-DELETE-METHOD, and *methodname-METHOD*).

These copybooks are used to manage connectivity to the CICS COBOL Component executing in the MAES region and to invoke the functions defined in the component. There are three connectivity protocols that can be used to manage the CICS COBOL component. To see example CICS COBOL programs demonstrating each protocol, look at the Aion-supplied sample programs CLCICCB (Init/Cleanup), CLCICCB2 (Resume/Suspend) and CLCICCB3 (Acquire/Release). Contact your systems programmer to locate the Aion sample programs.

To use the CICS COBOL component

1. To access the CICS COBOL component, your COBOL client program populates the required fields in the copybook area associated with the desired functionality and calls the CICS COBOL generic client DLL RECCOB*nn*. For example, a call to create an instance of the OBOOLEANS class would look like this:

   ```
   SET OBOOLEANS-CREATE-EIB TO ADDRESS OF DFHEIBLK.
   MOVE MAESCONN TO OBOOLEANS-CREATE-MAES.
   CALL 'RECCOBnn' USING OBOOLEANS-CREATE-METHOD.
   ```

2. Copy *<aionhlq>.appname*.COBOL(*appname*) to a copy library specified in the SYSLIB DD of the compile JCL. This library should be fixed block 80. Alternatively, you can point your SYSLIB directly to the dataset containing the generated member, <aionhlq>.appname.COBOL.

3. Place the client COBOL program in a source library defined as fixed block 80. Modify the client program appropriately, using the generated sample code as a guide. Supply the correct value for the MAES connection name (MAESCONN).

4. Preprocess the client COBOL program, and compile the output from the preprocessor.
5. The prelink of the COBOL client program should include an entry for SDSLIB(RECCOB*nn*) when running the Aion app in Suspend/Resume mode. When running the Aion app in Non-suspend mode, the prelink of the COBOL client program should include an entry for SDSLIB(RECCON*nn*).

   For example, for Suspend/Resume mode, include the following statements in the input to the prelink step:

   ```
   INCLUDE REOBJ(cobolclientname)
   INCLUDE SDSLIB(RECCOBnn) [for Suspend/Resume]
   NAME cobolclientname(R)
   ```

   Specify appropriate DD statements for each of the INCLUDE statements. For the INCLUDE SDSLIB statement, specify a DD statement pointing to the Aion SDS delivered with the Aion installation.

   Keep in mind that client programs using RESUME, SUSPEND, ACQUIRE or RELEASE functions must be executed in Non-Suspend mode. Attempts to execute these functions in Suspend/Resume mode will fail with a Return Code value of 4.

   Important! CICS client programs using the RESUME and SUSPEND or ACQUIRE and RELEASE functions must ensure that the APPC conversation ID (EIBRSRCE, an 8-byte character field in the EIB), remains constant for all function calls between the RESUME and SUSPEND or ACQUIRE and RELEASE calls. The EIBRSRCE field contains the ID of the APPC conversation with MAES and is overridden by CICS with the conversation ID for the user's terminal. Unless a client program makes all its component calls between terminal SENDs, the EIBRSRCE value must be saved prior to each terminal I/O and restored afterwards. Otherwise, the next method call fails because the connection to MAES cannot be identified correctly.
6. Define the following entities to CICS as programs:
   - The COBOL client program
   - The CICS COBOL client DLLs, RECCOB*nn* and RECCON*nn*
   - The CICS communication DLLs, RECWS*nn* and RECWN*nn*
   - The Aion system DLLs, RESYS*nn*, REUTIL*nn*, and RECOMD*nn*
7. The libraries containing the COBOL client program and the Aion DLLs must be specified in the DFHRPL DD concatenation of the CICS startup JCL.
8. Define a transaction to invoke the COBOL client program.

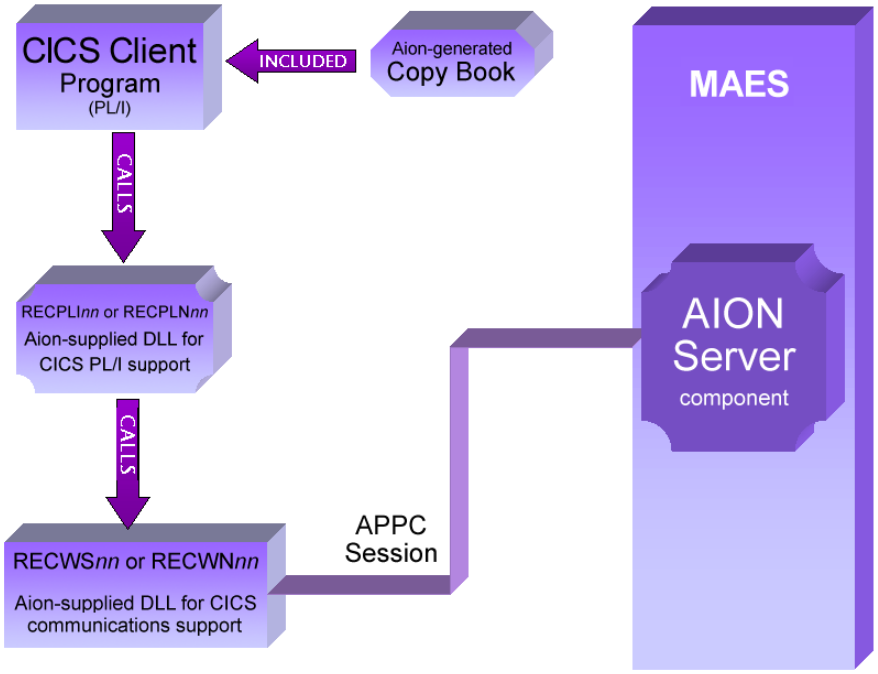
To run

1. Start MAES.
2. Start CICS.
   a. Invoke the transaction for the COBOL client program.

Generate a CICS PL/I Component

Use the PLI-CICS interface layer to generate sample PL/I data definitions describing the parameters and return values for each individual function within the component. Aion also generates sample PL/I code to call each of those functions.

The following figure illustrates a CICS PL/I client program communicating with an Aion server component in MAES:



In the illustration, RECPLI*nn* is the Suspend version. The Non-Suspend version is named RECPLN*nn*. The Aion calls in the PL/I client program always specify RECPLI*nn*. The SDS file linked with the PL/I client determines whether RECPLI*nn* or RECPLN*nn* actually is invoked.

Note: For r11.0 the module suffix (nn) is B0.

More Information:

Build and Manage Aion Applications (see page 71)

Specify the CICS PL/I Build Directives

There are three ways to specify build directives and generate the required client and server components:

- Use Batch Build with the parameter WRAPPER=PLI-CICS. Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.
- Remote Build capabilities on the Windows workstation

The following code fragments are produced when generating the CICS PL/I component:

Generated	Purpose	Location
Sample PL/I Code	Cut and paste into the PL/I program for use with the copybook.	*<aionhlq>.appname*.PL1(SAMPLE)
Copybook	Include in the PL/I program (using the COPY statement). The copybook contains data definition statements comprising one communications area for each call.	*<aionhlq>.appname*.PL1(*appname*)

Note: The generated PL/I external definition files include strings containing the names of exported classes and methods. The processing of these strings in MAES is case sensitive. When editing generated PL/I copybooks, do not modify the case of string values containing class and method names.

To generate field names, CA Aion BRE uses the names of an application's class, method, and argument, concatenated with other text.

For more information, see the *CA Aion BRE Product Guide*.

Use the CICS PL/I Component

The generated copybook contains a complete communications area for each method (CM_*appname*_INIT, CM_*appname*_SUSPEND, CM_*appname*_RESUME, CM_*appname*_ACQUIRE, CM_*appname*_RELEASE, CM_*appname*_CLEANUP, CM_*objname*_CREATE, CM_*objname*-DELETE, and CM_*objname*_*methodname*).

These copybooks are used to manage connectivity to the CICS PL/I Component executing in the MAES region and to invoke the functions defined in the component. There are three connectivity protocols, which can be used to manage the CICS PL/I component. To see example CICS PL/I programs demonstrating each protocol, look at the Aion-supplied sample programs CLCICPL (Init/Cleanup), CLCICPL2 (Resume/Suspend) and CLCICPL3 (Acquire/Release). For more information refer to <userhlq>.SAMPLES.SOURCE for specific CICS PL/I examples.

.To use the CICS PL/I component

1. To access the CICS PL/I component, your PL/I client program populates the required fields in the copybook area associated with the desired functionality and calls the CICS PL/I generic client DLL RECPLI*nn*. For example, a call to create an instance of the OBOOLEANS class would look like this:

   ```
   OBOOLEANS_CREATE_MAESNAME = MAESCONN;
   OBOOLEANS_CREATE_EIB      = ADDR(DFHEIBLK);
   CALL 'RECPLInn' (CM_OBOOLEANS_CREATE);
   ```

2. Copy <aioinhlq>.*appname*.PL1(*appname*) to a copy library specified in the SYSLIB DD of the compile JCL. This library should be fixed block 80. Alternatively, you can point your SYSLIB directly to the dataset containing the generated member, <aioinhlq>.*appname*.PL1.

3. Place the client PL/I program in a source library defined as fixed block 80. Modify the client program appropriately, using the generated sample code as a guide. Supply the correct value for the MAES connection name (MAESCONN).

4. Preprocess the client PL/I program, and compile the output from the preprocessor.
5. The prelink of the PL/I client program should include an entry for SDSLIB(RECPLI*nn*) when running the Aion application in Suspend/Resume mode. When running the Aion app in Non-suspend mode, the prelink of the PL/I client program should include an entry for SDSLIB(RECPLN*nn*).

   Note: The nn suffixes above are associated with the Aion BRE version you are using. For r11.0 the nn suffix is B0.

   For example, for Suspend/Resume mode, include the following statements in the input to the prelink step:

   ```
   INCLUDE REOBJ(pliclientname)
   INCLUDE SDSLIB(RECPLInn) [for Suspend/Resume]
   NAME pliclientname(R)
   ```

   Specify appropriate DD statements for each of the INCLUDE statements. For the INCLUDE SDSLIB statement, specify a DD statement pointing to the Aion SDS delivered with the Aion installation.

   Keep in mind that client programs using SUSPEND, RESUME, ACQUIRE or RELEASE functions must be executed in Non-Suspend mode. Attempts to execute these functions in Suspend/Resume mode will fail with a Return Code value of 4.

   Important! CICS client programs using the RESUME and SUSPEND or ACQUIRE and RELEASE functions must ensure that the APPC conversation ID (EIBRSRCE, an 8-byte character field in the EIB), remains constant for all function calls between the RESUME and SUSPEND or ACQUIRE and RELEASE calls. The EIBRSRCE field contains the ID of the APPC conversation with MAES and is overridden by CICS with the conversation ID for the user's terminal. Unless a client program makes all its component calls between terminal SENDs, the EIBRSRCE value must be saved prior to each terminal I/O and restored afterwards. Otherwise, the next method call fails because the connection to MAES cannot be identified correctly.
6. The following entities must be defined to CICS as programs:
   - The PL/I client program
   - The CICS PL/I client DLLs, RECPLI*nn* and RECPLN*nn*
   - The CICS communication DLLs, RECWS*nn* and RECWN*nn*
   - The Aion system DLLs, RESYS*nn*, REUTIL*nn*, and RECOMD*nn*

7. The libraries containing the PL/I client program and the Aion DLLs must be specified in the DFHRPL DD concatenation of the CICS startup JCL.
8. Define a transaction to invoke the PL/I client program.

To run

1. Start MAES.
2. Start CICS.
3. Invoke the transaction for the PL/I client program.

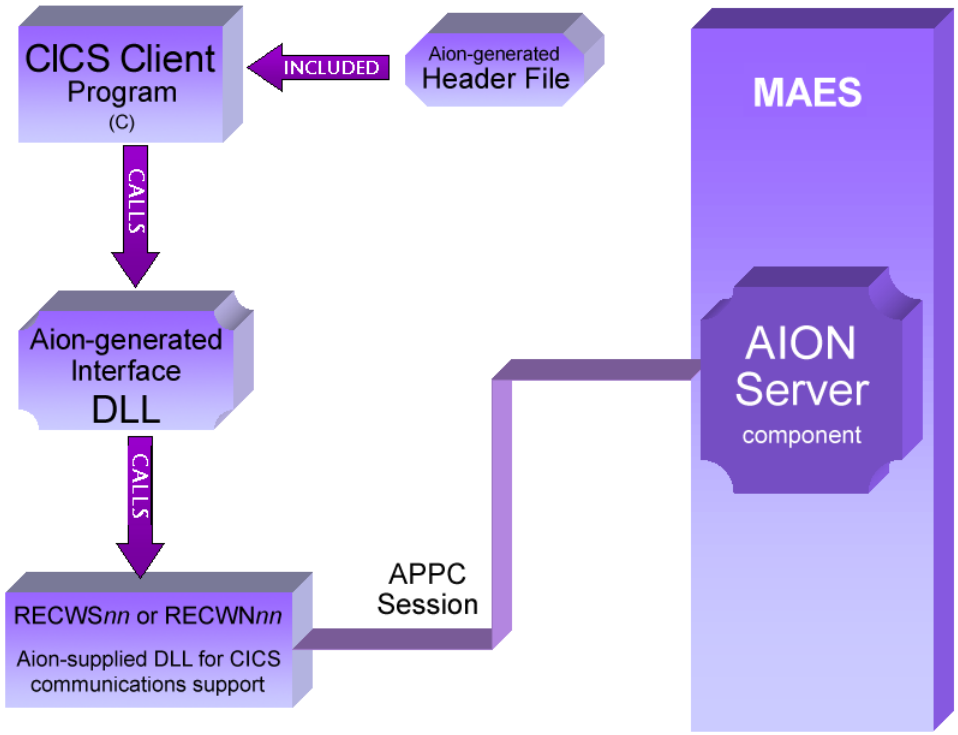
More Information:

Supported Data Types in Arguments (see page 115)

Generate a CICS C Component

Use the Aion C-CICS interface layer to generate a client DLL that exposes all the public functions in exported classes in the component, and ships these functions and their parameters to and from the server-side component for execution.

The following figure illustrates a CICS C client program communicating with an Aion server component in MAES.



Specify the CICS C Build Directives

To specify build directives and generate the required client and server components, use one of the following methods:

- Batch Build with the parameter WRAPPER=C-CICS

  Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.

- Remote Build capabilities on the Windows workstation

Important! Your client program should save the APPC conversation ID (EIBRSRCE, an 8-byte character field in the EIB) before each terminal SEND, and then restore it afterwards. This field contains the ID of the APPC conversation with MAES, and is overridden by CICS with the conversation ID for the user's terminal. Unless the client saves and restores the conversation ID, the next method call fails because the connection to MAES cannot be identified correctly.

CA Aion BRE compiles the server application, placing the client and server DLLs in the specified libraries. The header file to be included in the client program is placed in the *<aionhlq>.appname*.BIN2 data set.

<aionhlq>.appname.APP

The name of the Aion application.

The generated client-support DLL exposes the Aion application's public methods (with modified names), as well as the Aion-provided methods: *appname*_Init, *appname*_Suspend, *appname*_Resume, *appname*_Acquire, *appname*_Release, *appname*_Cleanup, *objname*_Create, and *objname*_Delete.

Note: The generated client DLL depends on other DLLs supplied with Aion; RECWS*nn* (the Suspend version of the CICS communication DLL), and RECWN*nn* (the Non-Suspend version of the CICS communication DLL).

For more information, see the *CA Aion BRE Product Guide*.

More Information:

Build and Manage Aion Applications (see page 71)

Use the CICS C Component

To use the CICS C component

1. Route the generated client DLL to a CICS DFHRPL load library.

   The C client uses this client DLL to access the Aion component server's external public methods (exposed in the Aion-generated client DLL) and the Aion-provided create, delete, initialization, suspend, resume, acquire, release and cleanup methods.

   The C user application accesses the client-support DLL in one of the following ways:

   – Uses function pointers via DLLLoad and DLLQueryFLinks using a side deck (SDS), allowing the program to call the functions directly. To do this, specify linkage editor statements as shown:

   ```
   INCLUDE OBJECT(clientname)
   INCLUDE BUILD2(SDS)
   INCLUDE AIONSDS(RESYSnn)
   NAME clientname(R)
   ```

   where:

   BUILD2 is a DD statement pointing to *<aionhlq>.appname*.BUILD2, which contains a member named SDS. This SDS member was generated when the client DLL was built.

   AIONSDS is a DD statement pointing to the SDS file created during the Aion installation.

2. Include the Aion-generated header file in the C client application. The header file is located in *<aionhlq>.appname*.BIN2(*appname*).

3. In the client application, specify a pointer to the client EIB and an appropriate value for the MAES connection name (MAESCONN) in the *appname*_Init call.

   The minimum sequence required to invoke a method directly from a C program is:

   ```
   appname_Init();
   classname_Create();
   classname_methodname();
   classname_Delete();    [Can be skipped since the app is being terminated]
   appname_Cleanup();
   ```

Init and Cleanup are not the only connectivity option. There are three protocols available to manage connectivity to the CICS C Component executing in the MAES region. To see example CICS C programs demonstrating each protocol, look at the Aion-supplied sample programs CLCICC (Init/Cleanup), CLCICC2 (Resume/Suspend) and CLCICC3 (Acquire/Release). Contact your systems programmer to locate the Aion sample programs.

These (and all other) remote function calls invoke CICS APPC functions in RECWS*nn or RECWNnn* to communicate with the Aion server component in MAES. These wrapper functions are described as follows:

appname_Init

*appname*_Init establishes a connection (APPC session) to MAES, and sends a START command to MAES to create a new subtask and start the component server program.

Init then sends a command message to the component server requesting the server DLL be loaded.

classname_Create

*classname*_Create is generated for each class exported by the component; *classname*_Create creates an instance of *classname* and returns a handle to that instance, which is used for all method calls.

classname_methodname

Classname_methodname is generated for each instance method in each class exported by the component; *classname_methodname* invokes the corresponding functionality in the server component.

classname_Delete

*classname*_Delete is generated for each class exported by the component; *classname*_Delete destroys the instance created by *classname*_Create.

appname_Cleanup

*appname*_Cleanup terminates the component server in MAES by sending an ENDCONS command message

This terminates the APPC session.

4. Define the following entities to CICS as programs:
   - The C client program
   - The CICS C client DLL
   - The CICS communication DLLs, RECWS*nn* and RECWN*nn*
   - The Aion system DLLs, RESYS*nn*, REUTIL*nn*, and RECOMD*nn*
5. The libraries containing the C client program and the Aion DLLs must be specified in the DFHRPL DD concatenation of the CICS startup JCL.
6. Define a transaction to invoke the C client program.

To run

1. Start MAES.
2. Start CICS.
3. Invoke the CICS transaction associated with the C client program.

To Use Suspend/Resume

The client DLL calls the CICS communication DLL to bundle the call for delivery to MAES. The desired CICS communication DLL is specified using Build Directive options.

- **RECWNnn**-The CICS Non-Suspend communication DLL
- **RECWSnn** -The CICS Suspend/Resume communication DLL

The Suspend/Resume DLL, RECWS*nn*, automatically resumes and suspends the component for each method call. The Non-Suspend DLL, RECWN*nn*, only resumes or suspends the component if the client program invokes the *appname*_Resume and *appname*_Suspend methods. Programs that do not stay resident in memory for the entire sequence of method calls to the Aion component server must either use the Suspend/Resume DLL or invoke the *appname*_Suspend and *appname*_Resume operations as appropriate using the Non-Suspend DLL.

Note: For r11.0 the module suffix (nn) is B0.

Create and Use Aion Components for IMS Clients

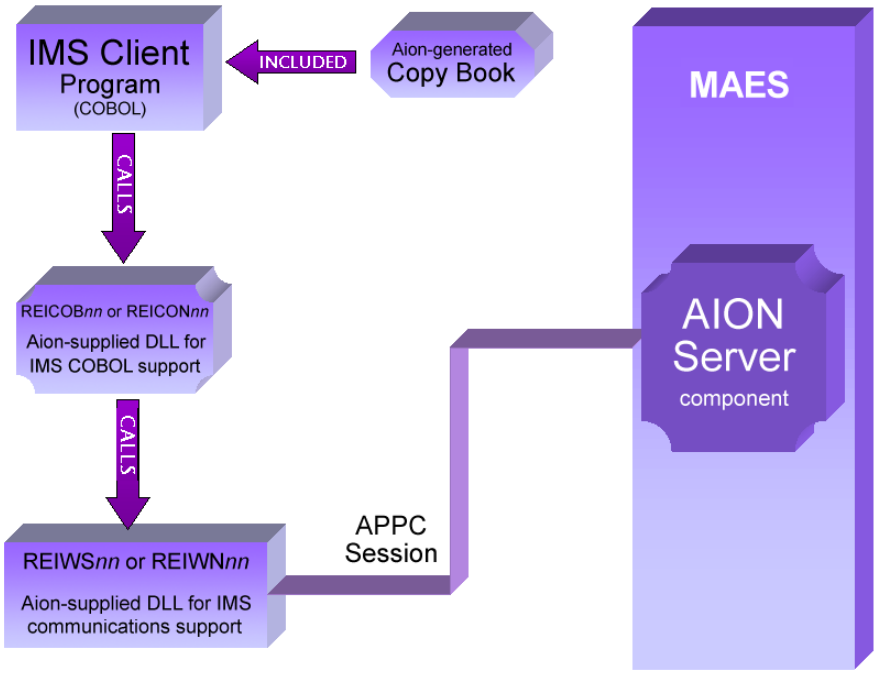
You can generate an Aion interface component to enable communication between an Aion server component and conversational or non-conversational IMS transaction programs (client programs) written in COBOL, PL/I, or C.

- To generate an IMS COBOL-compatible or IMS PL/I-compatible Aion component, build your Aion server application with a COBOL-IMS or PLI-IMS interface layer, respectively. This generates the server component itself. It also generates a copybook (file of data definition statements) to be copied into the data definition area of the COBOL or PL/I user program.
- To generate an IMS C-compatible Aion component, build your Aion server application with a C-IMS interface. This generates the Aion server component DLL. It also generates a header file and a client component DLL that provides access to all exported public functions in the Aion server component DLL.

Generate an IMS COBOL Component

Use the COBOL-IMS interface layer to generate sample COBOL data definitions describing the parameters and return values for each individual function within the component. CA Aion BRE also generates sample COBOL code to call each of those functions.

The following figure illustrates an IMS COBOL client program communicating with an Aion server component in MAES.



In the illustration, REICOB*nn* is the Suspend version. (The Non-Suspend version is named REICON*nn*. The Aion calls in the COBOL client program always specify REICOB*nn*. The SDS file linked with the COBOL client determines whether REICOB*nn* or REICON*nn* actually is invoked.

Note: For r11.0 the module suffix (nn) is B0.

More Information:

Build and Manage Aion Applications (see page 71)

Specify the IMS COBOL Build Directives

To specify build directives and generate the required client and server components, use one of the following methods:

- Batch Build with the parameter WRAPPER=COBOL-IMS

  Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.
- Remote Build capabilities on the Windows workstation

The following are produced when generating the IMS COBOL component:

Generated	Purpose	Location
Sample COBOL Code	Cut and paste into the COBOL program for use with the copybook.	<*aionhlq*>.*appname*.COBOL(SAMPLE)
Copybook	Include in the COBOL program (using the COPY statement). The copybook contains data definition statements comprising one communications area for each call.	<*aionhlq*>.*appname*.COBOL(*appname*)

Note: The generated COBOL external definition files include strings containing the names of exported classes and methods. The processing of these strings in MAES is case sensitive. When editing generated COBOL copybooks, do not modify the case of string values containing class and method names.

To generate field names, CA Aion BRE uses the names of an application's class, method and argument, concatenated with other text.

For more information, see the *CA Aion BRE Product Guide*.

Use the IMS COBOL Component

The generated copybook contains a complete communications area for each method (INIT-METHOD, SUSPEND-METHOD, RESUME-METHOD, ACQUIRE-METHOD, RELEASE-METHOD, CLEANUP-METHOD, *objname*-CREATE-METHOD, *objname*-DELETE-METHOD, and *methodname-METHOD*).

These copybooks are used to manage connectivity to the IMS COBOL Component executing in the MAES region and to invoke the functions defined in the component. There are three connectivity protocols, which can be used to manage the IMS COBOL component. To see example IMS COBOL programs demonstrating each protocol, look at the Aion-supplied sample programs CLIMSCB (Init/Cleanup), CLIMSCB2 (Resume/Suspend) and CLIMSCB3 (Acquire/Release). The samples can be found in <aionhlq>.SAMPLES.SOURCE.

To use the IMS COBOL component

1. To access the IMS COBOL component, your COBOL client program populates the required fields in the copybook area associated with the desired functionality and calls the IMS COBOL generic client DLL REICOB*nn*. For example, a call to create an instance of the OBOOLEANS class would look like this:

   ```
   SET OBOOLEANS-CREATE-PCB TO ADDRESS OF LNK-IOPCB
   MOVE MAESNAME            TO OBOOLEANS-CREATE-MAES
   MOVE LOCALLU             TO OBOOLEANS-CREATE-LOCALLU
   CALL 'REICOBnn' USING OBOOLEANS-CREATE-METHOD.
   ```

   Note: For r11.0 the module suffix (nn) is B0.

2. Copy *<aionhlq>.appname.*COBOL(*appname*) to a copy library specified in the SYSLIB DD of the compile JCL. This library should be fixed block 80. Alternatively, you can point your SYSLIB directly to the dataset containing the generated member, *<aionhlq>.appname*.COBOL.

3. Place the client COBOL program in a source library defined as fixed block 80. Modify the client program appropriately, using the generated sample code as a guide. Supply the correct values for the MAES and LOCALLU node names.

4. The prelink of the COBOL client program should include an entry for SDSLIB(REICOB*nn*) when running the Aion app in Suspend/Resume mode. When running the Aion app in Non-suspend mode, the prelink of the COBOL client program should include an entry for SDSLIB(REICON*nn*).

   For example, for Suspend/Resume mode, include the following statements in the input to the prelink step:

   ```
   INCLUDE REOBJ(cobolclientname)
   INCLUDE SDSLIB(REICOBnn) [for Suspend/Resume]
   NAME cobolclientname(R)
   ```

   Specify appropriate DD statements for each of the INCLUDE statements. For the INCLUDE SDSLIB statement, specify a DD statement pointing to the Aion SDS delivered with the Aion installation.

5. The libraries containing the COBOL client program and the Aion DLLs (REICOB*nn*, REICON*nn,* REIWS*nn*, REIWN*nn,* RESYS*nn*, REUTIL*nn*, and RECOMD*nn*) must be specified in the STEPLIB DD concatenation of the IMS Message Processing Region JCL.

6. Define a transaction to invoke the COBOL client program.

To run

1. Start MAES.
2. Start the IMS message processing region.
3. Invoke the transaction for the COBOL client program.

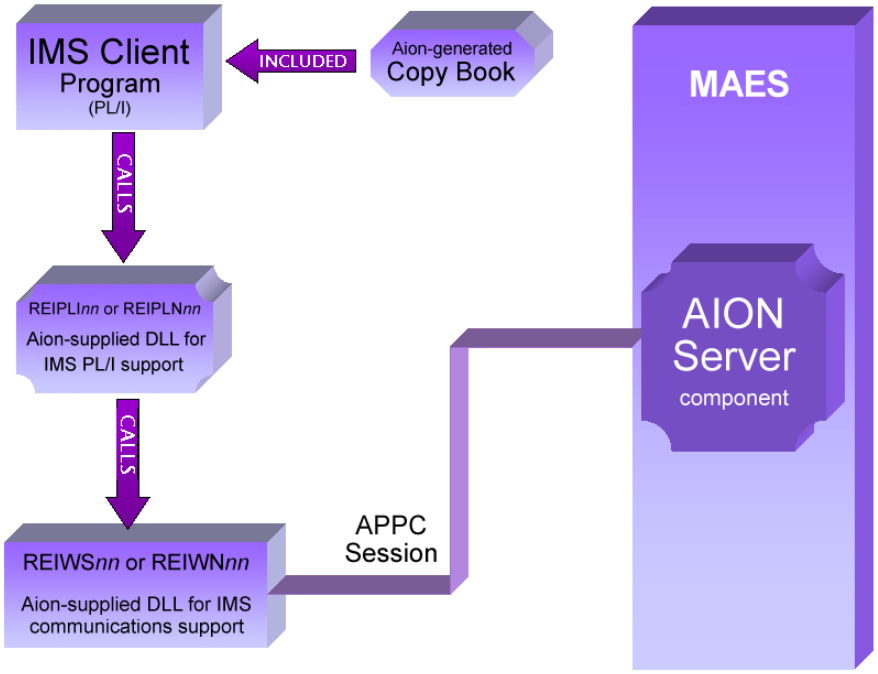
More Information

Supported Data Types in Arguments (see page 115)

Generate an IMS PL/I Component

Use the PLI-IMS interface layer to generate sample PL/I data definitions describing the parameters and return values for each individual function within the component. Aion also generates sample PL/I code to call each of those functions.

The following figure illustrates an IMS PL/I client program communicating with an Aion server component in MAES.



In the illustration, REIPLI*nn* is the Suspend version. (The Non-Suspend version is named REIPLN*nn*. The Aion calls in the PL/I client program always specify REIPLI*nn*. The SDS file linked with the PL/I client determines whether REIPLI*nn* or REIPLN*nn* actually is invoked.

Note: For r11.0 the module suffix (nn) is B0.

Specify the IMS PL/I Build Directives

To specify build directives and generate the required client and server components, use one of the following methods:

- Batch Build with the parameter WRAPPER=PLI-IMS

  Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.

- Remote Build capabilities on the Windows workstation

The following code fragments are produced when generating the IMS PL/I component:

Generated	Purpose	Location
Sample PL/I Code	Cut and paste into the PL/I program for use with the copybook.	<*aionhlq*>.*appname*.PL1(SAMPLE)
Copybook	Include in the PL/I program (using the COPY statement). The copybook contains data definition statements comprising one communications area for each call.	<*aionhlq*>.*appname*.PL1(*appname*)

Note: The generated PL/I external definition files include strings containing the names of exported classes and methods. The processing of these strings in MAES is case sensitive. When editing generated PL/I copybooks, do not modify the case of string values containing class and method names.

To generate field names, CA Aion BRE uses the names of an application's class, method, and argument, concatenated with other text.

For more information, see the *CA Aion BRE Product Guide*.

More Information:

Build and Manage Aion Applications (see page 71)

Use the IMS PL/I Component

The generated copybook contains a complete communications area for each method (CM_*appname*_INIT, CM_*appname*_SUSPEND, CM_*appname*_RESUME, CM_*appname*_ACQUIRE, CM_*appname*_RELEASE, CM_*appname*_CLEANUP, CM_*objname*_CREATE, CM_*objname*-DELETE, and CM_*objname*_*methodname*).

These copybooks are used to manage connectivity to the IMS PL/I Component executing in the MAES region and to invoke the functions defined in the component. There are three connectivity protocols, which can be used to manage the IMS PL/I component. To see example IMS PL/I programs demonstrating each protocol, look at the Aion-supplied sample programs CLIMSPL (Init/Cleanup), CLIMSPL2 (Resume/Suspend) and CLIMSPL3 (Acquire/Release). Please refer to <userhlq>.SAMPLES.SOURCE for specific PL/I examples.

To use the IMS PL/I component

The nn suffixes below are associated with the Aion BRE version you are using. For r11.0 the nn suffix is B0.

1. To access the IMS PL/I component, your PL/I client program populates the required fields in the copybook area associated with the desired functionality and calls the IMS PL/I generic client DLL REIPLI*nn*. For example, a call to create an instance of the OBOOLEANS class would look like this:

   ```
   OBOOLEANS_CREATE_MAESNAME = MAESNAME;
   OBOOLEANS_CREATE_LOCALLU  =  LOCALLU;
   OBOOLEANS_CREATE_PCB      =  PCB1;
   CALL 'REIPLInn' (CM_OBOOLEANS_CREATE);
   ```

   To see an example, look at the Aion-supplied sample program named CLIMSPL. Contact your systems programmer to locate the Aion sample programs.

2. Copy <*aionhlq*>.*appname.*PL1(*appname*) to a copy library specified in the SYSLIB DD of the compile JCL. This library should be fixed block 80. Alternatively, you can point your SYSLIB directly to the dataset containing the generated member, <aionhlq>.*appname*.PL1.

3. Place the client PL/I program in a source library defined as fixed block 80. Modify the client program appropriately, using the generated sample code as a guide. Supply the correct values for the MAES and LOCALLU node names.

4. The prelink of the PL/I client program should include an entry for SDSLIB(REIPLI*nn*) when running the Aion app in Suspend/Resume mode. When running the Aion app in Non-suspend mode, the prelink of the PL/I client program should include an entry for SDSLIB(REIPLN*nn*).

   For example, for Suspend/Resume mode, include the following statements in the input to the prelink step:

   ```
   INCLUDE REOBJ(pliclientname)
   INCLUDE SDSLIB(REIPLInn) [for Suspend/Resume]
   NAME pliclientname(R)
   ```

   Specify appropriate DD statements for each of the INCLUDE statements. For the INCLUDE SDSLIB statement, specify a DD statement pointing to the Aion SDS delivered with the Aion installation.

5. The following entities must be in the IMS message region STEPLIB:
   - The PL/I client program
   - The IMS PL/I client DLLs, REIPLI*nn* and REIPLN*nn*
   - The IMS communication DLLs, REIWS*nn* and REIWN*nn*
   - The Aion system DLLs, RESYS*nn*, REUTIL*nn*, and RECOMD*nn*
6. Define a transaction to invoke the PL/I client program.

To run

1. Start MAES.
2. Start the IMS message region.
3. Invoke the transaction for the PL/I client program.

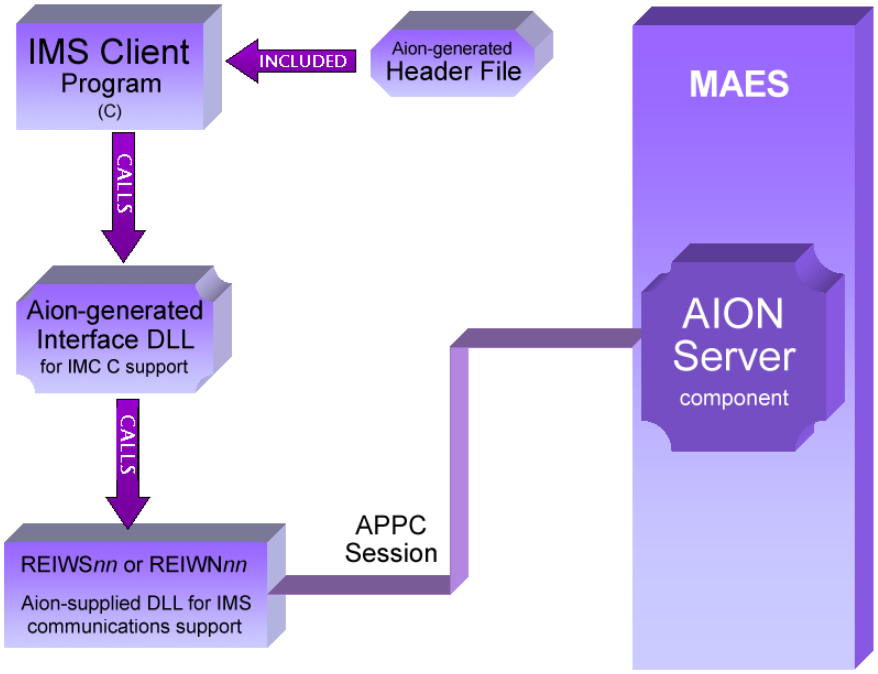
More Information:

Supported Data Types in Arguments (see page 115)

Generate an IMS C Component

Use the Aion C-IMS interface layer to generate a client DLL which exposes all the public functions in exported classes in the component, and ships these functions and their parameters to and from the server-side component for execution.

The following figure illustrates an IMS C client program communicating with an Aion server component in MAES.



Specify the IMS C Build Directives

To specify build directives and generate the required client and server components, use one of the following methods:

- Batch Build with the parameter WRAPPER=C-IMS

  Optionally, you can specify CONVERSATIONAL to suppress Suspend/Resume logic.

- Remote Build capabilities on the Windows workstation

CA Aion BRE compiles the server application, placing the client and server DLLs in the specified libraries. The header file to be included in the client program is placed in the <*aionhlq*>.*appname*.BIN2 data set (where <*aionhlq*>.*appname*.APP is the name of the Aion application).

- The generated client-support DLL exposes the Aion application's public methods (with modified names), as well as the Aion-provided methods: appname_Init, appname_Suspend, appname_Resume, appname_Acquire, appname_Release, appname_Cleanup, objname_Create, and objname_Delete.

  Note: The generated client DLL depends on other DLLs supplied with Aion; REIWSnn (the Suspend version of the IMS communication DLL), and REIWNnn (the Non-Suspend version of the IMS communication DLL).

  Note: The nn suffixes above are associated with the Aion BRE version you are using. For r11.0 the nn suffix is B0.

  For more information, see the *CA Aion BRE Product Guide*.

More Information:

Build and Manage Aion Applications (see page 71)

Use the IMS C Component

To use the IMS C component

1. Route the generated client DLL to a load library available to the IMS message region.
2. The C client uses this client DLL to access the Aion component server's external public methods (exposed in the Aion-generated client DLL) and the Aion-provided create, delete, initialization, suspend, resume, acquire, release and cleanup methods.
3. The C user application accesses the client-support DLL in one of the following way:
4. Function pointers using DLLLoad and DLLQueryFn.Links using a side deck (SDS), allowing the program to call the functions directly. To do this, specify linkage editor statements as shown:

   ```
   INCLUDE OBJECT(clientname)
   INCLUDE BUILD2(SDS)
   INCLUDE AIONSDS(RESYSnn)
   NAME clientname(R)
   ```

   Where BUILD2 is a DD statement pointing to <*aionhlq*>.*appname*.BUILD2, which contains a member named SDS. This SDS member was generated when the client DLL was built. AIONSDS is a DD statement pointing to the SDS file created during the Aion installation.

5. Include the Aion-generated header file in the C client application. The header file is located in *<aionhlq>.appname*.BIN2(*appname*).

6. In the client application, specify a pointer to the I/O PCB and an appropriate value for the MAES and LOCALLU node names in the *appname*_Init call.

   The minimum sequence required to invoke a method directly from a C program is:

   ```
   appname_Init();
   classname_Create();
   classname_methodname();
   classname_Delete();    [Can be skipped since the app is being terminated]
   appname_Cleanup();
   ```

   Init and Cleanup are not the only connectivity option. There are three protocols available to manage connectivity to the IMS C Component executing in the MAES region. To see example IMS C programs demonstrating each protocol, look at the Aion-supplied sample programs CLIMSC (Init/Cleanup), CIMSC2 (Resume/Suspend) and CLIMSC3 (Acquire/Release). Contact your systems programmer to locate the Aion sample programs.

   These (and all other) remote function calls invoke APPC functions in REIWS*nn or REIWNnn* to communicate with the Aion server component in MAES. These wrapper functions are described as follows:

   appname_Init

   *appname*_Init establishes a connection (APPC session) to MAES, and sends a START command to MAES to create a new subtask and start the component server program. Init then sends a command message to the component server requesting the server DLL be loaded

   classname_Create

   *classname*_Create is generated for each class exported by the component; *classname*_Create creates an instance of *classname* and returns a handle to that instance, which is used for all method calls.

   classname_methodname

   Classname_methodname is generated for each instance method in each class exported by the component; *Classname_methodname* invokes the corresponding functionality in the server component.

   classname_Delete

   *classname*_Delete is generated for each class exported by the component; *classname*_Delete destroys the instance created by *classname*_Create.

appname_Cleanup

*appname*_Cleanup terminates the component server in MAES by sending an ENDCONS command message

This terminates the APPC session.

7. The following entities must be in the IMS message region STEPLIB:
   - The C client program
   - The IMS C client DLL
   - The IMS communication DLLs, REIWS*nn* and REIWN*nn*
   - The Aion system DLLs, RESYS*nn*, REUTIL*nn*, and RECOMD*nn*

   Note: For r11.0 the module suffix (nn) is B0
8. Define a transaction to invoke the C client program.

To run

1. Start MAES.
2. Start the IMS message region.
3. Invoke the IMS transaction associated with the C client program.

To Use Suspend/Resume

The client DLL calls the IMS communication DLL to bundle the call for delivery to MAES. The desired IMS communication DLL is specified using Build Directive options.

- **REIWNnn**-The IMS Non-Suspend communication DLL
- **REIWSnn** -The IMS Suspend/Resume communication DLL

The Suspend/Resume DLL, REIWS*nn*, automatically resumes and suspends the component for each method call. The Non-Suspend DLL, REIWN*nn*, only resumes or suspends the component if the client program invokes the *appname*_Resume and *appname*_Suspend methods. Programs that do not stay resident in memory for the entire sequence of method calls to the Aion component server must either use the Suspend/Resume DLL or invoke the *appname*_Suspend and *appname*_Resume operations as appropriate using the Non-Suspend DLL.

Note: For r11.0 the module suffix (nn) is B0.

More Information:

Specify the IMS C Build Directives (see page 138)

Create and Use Aion Components for TCP/IP Clients

The TCP/IP interface exposes the TCP/IP API to a PC client application so that the client can communicate through this API with an Aion component running under MAES. Aion application code is automatically translated into C code that is used to build a client DLL. This DLL, TR*x*IP.DLL (where *x* is 'C' for the CICS client and 'B' for the IMS/batch client), exposes the functions that the client application uses to communicate with the Aion component.

CA Aion BRE also generates a generic calling model, RExIPnn (where *x* is as above), that is called by the client TR*x*IP DLL to perform the TCP/IP communication with MAES. This second module packages the data into standard messages, sends those messages to the TCP/IP server in MAES and processes the return message through the client DLL to the client application.

Note: For r11.0 the module name is RExIPB0.

Note: Invocation of the TCP/IP API is accomplished through synchronous calls.

The TCP/IP API

Additional batch build wrapper options are available for TCP/IP clients. "WRAPPER=TCPIP" generates client code for building a client DLL for TCP/IP. For CICS clients the additional CICSIP keyword must be specified.

To assist you with understanding the TCP/IP API please refer to the CLIMSTP example which is a C++ program that interacts via TCP/IP with the ARGSERV.APP running in MAES. Details regarding the CLIMSTP example are described in <aionhlq>.SAMPLES.README(TCPIP).

The following API is provided to the customer in order for the customer application to invoke the Aion component.

- appname_Connect(host_name, port_number) : returnCode
  - Establishes the connection to the TCP/IP Server, where *appname* is the name of an Aion application.
  - Arguments are NULL terminated strings.
    - *host_name* can be in decimal dot notation, for example, 123.456.789, or a standard URL format, for example, myhost.server.net. When using IPv6 communications an explicit host name uses a colon delimited format; for example: 000a::b:0000:1111:2222
    - *port_number* is the port number assigned to the TCP/IP Server in the MAES address space.
  - The call returns 0 if it is successful. A non-zero return indicates an error. The following are the possible error returns:
    - 110 Unable to locate named host. Most likely cause is a bad host name.
    - 120 Unable to establish connection with host. Most likely causes are bad port number or server not active.
    - 130 Connection with host has been lost.
    - 140 Error message received from server. Text is in SYSOUT and on server log

- classname_Create(name, errorreturn) : val_InPtr
  - Creates an instance of the Aion class *classname*.
  - Arguments are:
    - *name* is a NULL terminated string to be used as the name of the instance. If this argument is NULL default names will be used
    - *errorreturn* is a pointer to an integer which is used to indicate a communication error. Possible errors are 130 and 140 as described previously.
  - The method returns an instance handle (val_InPtr). If the returned instance handle is zero the instance could not be created, error information should be available on the server.
- classname_methodname(handle, errorreturn, ...)
  - Invokes method *methodname* on Aion class *classname*.
  - Arguments are:
    - *handle* must be the instance handle returned by the *classname*_Create call.
    - *errorreturn* is a pointer to an integer which is used to indicate a communication error. Possible errors are 130 and 140 as described previously.
    - The remaining arguments (. . .) are those defined by the method being invoked. The format of these arguments is described in the "TCP/IP argument format" sections that follow.
- classname_Delete(handle, errorreturn)
  - Deletes the specific instance of Aion class *classname*.
  - Arguments are:
    - *handle* must be the instance handle returned by the *classname*_Create call. The instance is deleted
    - *errorreturn* is a pointer to an integer which is used to indicate a communication error. Possible errors are 130 and 140 as described previously.
- *appname*_Disconnect() : returnCode
  - Closes the connection between the client and the server, where *appname* is the name of an Aion application.
  - The call returns 0 (zero). Any errors are ignored and the connection is closed.

TCP/IP Argument Formats

Aion BRE supports two message formats for communicating with application methods - platform specific, and platform neutral.

TCP/IP Platform Specific Message Format

A platform specific format transmits and receives numbers and length values, character strings, dates, times, and real numbers in native formats. MAES implicitly converts these values to an internal z/OS format. For example, characters within text strings are converted from Ascii to EBCDIC Code Page 1047. Response values are also in native formats, with associated implicit conversion by MAES from internal z/OS formats.

The platform specific message format is restricted in two ways. Text and binary string values can't be longer than 64 kilobytes (65536). In addition, the total size of a message that is sent to invoke a method, and the associated length of the response can't be longer than 64 kilobytes (65536).

CLIMSTP.CPP Example

The CLIMSTP example program can be found in the <aionhlq>.SAMPLES.SOURCE dataset. It uses platform specific messages to communicate with the ARGSERV application in the MAES environment. Usage information for the example can be found in the TCPIP member of the <aionhlq>.SAMPLES.README dataset.

Message Prefix

Messages in platform specific format have a four byte message prefix

Offset	Size	Description
+0	2	Length of entire message (in native format)
+2	1	Request/response type 0x00 REQUEST 0x40 REQUEST 0x80 RESPONSE
+3	1	Client platform type 0x00 Win32s 0x01 Win32 0x02 Windows NT, XP 0x03 z/OS batch or TSO 0x04 UNIX/Linux 0x05 MAES subtask 0x06 TS) Foreground

Message Value Types

DLL names, function (method) names, parameters, and result values are preceded by a one byte message type:

Value	Description
0	Void (used for functions that do not return a value)
1	Integer
2	String
3	Real
4	Binary string
5	Boolean
6	Time
7	Date
8	Function
9	Dllname
0x40	Output parameter modifier (OR'd with value type)

Message Type 9 - DLL Load Request

DLL load requests are used to activate Aion BRE applications (KBs). The returned value is a numeric DLL instance handle.

(Message prefix, as described previously)

Offset	Size	Descriptio
+4	1	0x09
+5	2	length of DLL name nnn .. 1 to 8 (in native format)
+7	nnn	DLL module name (NOT zero terminated)

Message Type 8 - Function Call

Function calls are used to invoke DLL (KB) methods.

(Message prefix, as described previously)

Offset	Size	Description
+4	1	0x08
+5	1	Return type 0=VOID 1=INT 2=STRING 3=REAL 5=BOOL 6=TIME
+6	1	#parameters
+7	1	(unused, for alignment only)
+8	2	Length of function name nnn (in native format)
+10	nnn	The function name (NOT zero terminated)
+10+nnn	1	type of parameter 1=INT 2=STRING 3=REAL
If the parameter is a *boolean* or an *integer*		
+10+nnn+1	4	numeric value (in native format)
If the parameter is a *string* or *binary string*:		
+10+nnn+1	2	string length nnn (in native format)
+10+nnn+1	nnn	string value (not zero terminated)
If the parameter is a *real*		
+10+nnn+1	8	real value (in native format) Note: A C "time_t"
If the parameter is a *date*		
+10+nnn+1	6	date value, 3 numeric values (in native format): Year month Day of month

Offset	Size	Description
If the parameter is a *time*		
+10+nnn+1	4	C time value (in native format)

Note: Subsequent parameters follow immediately afterwards without any alignment adjustments.

Special First Parameter for Function Calls

The first parameter of class create and delete method requests is the DLL instance handle that was returned by a DLL load request.

The first parameter of all other method requests is the instance handle that is returned by a class create method request.

Response Formats

Response messages contain method return values, optionally followed by a method's output parameters if it has any.

(Message prefix, as described previously)

Offset	Size	Description
+4	1	type of response value 1=INT 2=LPSTR 3=REAL
If the return value is a *void*		
+4	0	nothing is sent (output parameters proceed at offset +4)
If the return value is a *boolean* or an *integer*		
+5	4	numeric value (in native format)
If the return value is a *string* or *binary string*:		
+5	2	string length nnn (in native format)
+7	nnn	string value (not zero terminated)
If the return value is a *real*		
+5	8	real value (in native format)
If the return value is a *date*		
+5	6	date value, 3 numeric values (in native format):

Offset	Size	Description
		Year Month Day of month
If the return value is a *time*		
+5	4	real value (in native format) Note: A C "time_t"

Note: Subsequent output parameters follow immediately afterwards without any alignment adjustments. Output parameters are formatted identically to the return values above.

TCP/IP Platform Neutral Message Format

The platform neutral format transmits and receives numbers, lengths, and character strings in a universal network format. Dates and real numbers are sent in text format. There are four time formats. Character strings are transmitted using Unicode Text Format 8 (UTF-8) encodings. All character strings are concluded by a null terminator, except if the length is zero. Very large text and binary string values, up to 8 megabytes, can be sent. The total size of a message can be up to 2 gigabytes. All values are aligned on 4 byte boundaries. Values are padded on the right with zeroes if required for alignment reasons.

Neutral Message Prefix

Messages in platform neutral format have an eight byte message prefix.

Offset	Size	Description
+0	1	0x00
+1	1	0x00
+2	1	Request/response type 0x00 REQUEST 0x40 REQUEST 0x80 RESPONSE
+3	1	Maker 0x88 Platform neutral message indicator

Offset	Size	Description
+4	4	Length of subsequent data (in network byte order)
+8	8	64 bit session identifier (Zeroes for a CONNECT request)

Note: Subsequent data proceeds at offset 16.

Message Value Types

DLL names, function (method) names, parameters, and result values are preceded by a one byte message type:

Value	Description
0	Void (used for functions that do not return a value)
1	Integer
2	String
3	Real
4	Binary string
5	Boolean
6	Time
7	Date
8	Function
9	Dllname
21	Timespan
22	ISO time
23	Timeval
33	Error text in UTF-8
35	64-bit integer
36	UTF-8 string
0x40	Output parameter modifier (OR'd with value type)

Time Value Types

Aion BRE stores time values in an internal format that is a signed 31-bit value, that is A C "time t" . It is the number of seconds since 00:00:00 on January 1st, 1970 in local time (not GMT time). In Aion BRE applications a time value can be:

- A time of day
- A time span
- The time on a specific date

Four different message types are supported to assist client programs in the processing of time values

Type Time (6)

The time value is a string formatted as

```
[YYYYMMDDT]hhmmss[uuu]
```

This is the string format of a local time value. A 'T' character separates the optional leading date from the hour, minute, and second value. This format conforms with how pre-r11.0 time values were processed. Since the time is a local time, this format is problematic when the value is passed to/from clients that are operating in other time zones.

Example: 20081204T121954

12:19:54 on December 4th, 2008 - in local time

Type Timespan (21)

The time value is a string formatted as

```
[#daysT]hhmmss[uuu]
```

This is the string format of a time duration, and consequently is independent of local time/Greenwich time considerations. A 'T' character separates the optional #days from the hour, minute, and second value.

Example: 17T112233

17 days, 11 hours, 22 minutes, 33 seconds

Type Isotime (22)

The time value is a string formatted as

```
[YYYYMMDDT]hhmmss[uuu][(+|-)GMTdelta]
```

This is the string format of an ISO time value. A 'T' character separates the optional leading date from the hour, minute, and second value. This is a new r11.0 format. The GMT delta is the amount of hours and minutes the time differs from Greenwich time. A negative value indicates that the time is behind Greenwich.

Example 1: 20081204T121954-0500

12:19:54 on December 4th, 2008, five hours behind Greenwich (EST, UTC - 5 hours)

Example 2: 20081204T121954+0100

12:19:54 on December 4th, 2008, one hour ahead of Greenwich (CET, UTC +1 hour)

Type timeval (23)

The time value is a signed 31-bit integer . This format is identical to the platform specific time value. Since the time is a local time, this format is problematic when the value is passed to/from clients that are operating in other time zones.

Message type 36 - String Command Requests

String command requests are used to establish or disestablish connections, or to end a consultation.

Connect Request

A connect_call establishes a MAES connection and verifies that the userid and password are authorized to use MAES KBs.

MAES must be executing authorized to perform signon verification. If MAES is unauthorized the userid and password are accepted without verification.

TCP/IP socket communications use Secure Socket Layer (SSL) protocols, consequently the userid and password are implicitly encrypted when transmitted.

(Neutral message prefix, as described in the proceeding sections.)

Offset	Size	Description
+16	4	36

Offset	Size	Description
+20	4	7 - length of "CONNECT"
+24	7	"CONNECT" (null terminated)
+32	4	36
+36	4	uuu length of userid
+40	uuu	userid (null terminated)
+40+uuu+(a)	4	36
+40+uuu+(a)	4	ppp length of password
+40+uuu+(a)	ppp	password (null terminated)

Success response:

(Neutral message prefix, as described in the proceeding sections.)

Offset	Size	Description
+16	4	35
+20	8	64-bit integer session identifier (passed o all subsequent requests)

Error response:

(Neutral message prefix, as described in the proceeding sections.)

Offset	Size	Description	
+16	4	36 utf8 string	
+20	4	nnn	Error string length

Disconnect Request

A disconnect request discontinues a MAES connection.

Note: when a session is disconnected all associated dll_handles, create_handles, and init_handles are implicitly closed.

(Neutral message prefix, as described previously)

Offset	Size	Description
+16	4	36
+20	4	10 - length of "DISCONNECT"

Offset	Size	Description
+24	10	"DISCONNECT" (null terminated)

End Consultation Request

An endcons request concludes a consultation. The associated DLL is removed from memory.

Note: when an endcons request is performed all associated create_handles, and init_handles are implicitly closed.

(Neutral message prefix, as described previously)

Offset	Size	Description
+16	4	36
+20	4	7 - length of "ENDCONS"
+24	nnn	"ENDCONS" (null terminated)

Message Type 9 - DLL Load Request

DLL load requests are used to activate Aion BRE applications (KBs). The returned value is a numeric DLL instance handle.

(Neutral message prefix, as described previously)

Offset	Size	Description
+16	4	0x09
+20	4	length of DLL name nnn .. 1 to 8
+24	nnn	DLL module name (null terminated)

Message Type 8 - Function call

Function calls are used to invoke DLL (KB) methods.

(Neutral message prefix, as described previously)

Offset	Size	Description
+16	4	0x08
+20	4	Return type 0=VOID

Offset	Size	Description
		1=INT 2=STRING 3=REAL 5=BOOL 6=TIME
+24	4	#parameters
+28	4	Length of function name *mmm*
+32	*mmm*	The function name (null terminated)
+32+mmm+(a)	4	type of parameter 1=INT 2=STRING 3=REAL **Note**: (a) is an alignment padding length. All values are aligned on a 4-byte boundary. All alignment padding bytes are ignored on input, and are zeroes on output.
If the parameter is a *boolean* or an *integer*		
+32+mmm+(a)	4	numeric value (in network byte order)
If the parameter is a *string*		
+32+mmm+(a)	4	string length *nnn* (in network byte order)
+36+mmm+(a)	nnn	UTF-8 string value (null terminated)
If the parameter is a binary *string*		
+32+mmm+(a)	4	string length *nnn* (in network byte order)
+36+mmm+(a)	nnn	binary string value (not null terminated)
If the parameter is a *real*		
+32+mmm+(a)	4	string length *nnn* (in network byte order)
+36+mmm+(a)	8	real value (in string)
If the parameter is a *date*		
+32+mmm+(a)	4	string length *nnn* (in network byte order)
+36+mmm+(a)	8	date string: YYYYMMDD
If the parameter is a *time*		
+32+mmm+(a)	4	string length *nnn* (in network byte order)
+36+mmm+(a)	nnn	time string:

Offset	Size	Description
		`[YYYYMMDDT]hhmmss[uuu]` uuu is milliseconds The 'T' separates the optional YYYYMMDD date from the time value **Note**: this format is subject to difficulties between the client time zone and the MAES time zone
If the parameter is a *timespan*		
+32+mmm+(a)	4	string length nnn (in network byte order)
+36+mmm+(a)	nnn	time string: A UTF-8 string formatted as: #daysThhmmss[uuu] uuu is milliseconds. The 'T' separates the optional #days from the time value
If the parameter is a *isotime*		
+32+mmm+(a)	4	string length nnn (in network byte order)
+36+mmm+(a)	nnn	time string: A UTF-8 string formatted as: `YYYYMMSSThhmmss[uuu][(+|-)GMTdelta]` uuu is milliseconds The 'T' separates the optional YYYYMMDD date from the time value The GMTdelta is the difference from Greenwich example, -0500 indicates the time is 5 hours behind Greenwich
If the parameter is a *timeval*		
+32+mmm+(a)	4	numeric time value (in network byte order) **Note**: this format is subject to difficulties between the client time zone and the MAES time zone.

Note: Subsequent parameters follow immediately afterwards without any alignment adjustments.

Special First Parameter for Function Calls

The first parameter of class create and delete method requests is the DLL instance handle that was returned by a DLL load request.

The first parameter of all other method requests is the instance handle that is returned by a class create method request.

Response Formats

Response messages contain method return values, optionally followed by a method's output parameters if it has any.

(Neutral message prefix, as described previously)

Offset	Size	Description
+8	4	type of response value 0=VOID 1-INT 2=STRING 3=REAL
If the return value is a *void*		
+12	0	nothing is sent (output parameters proceed at offset +12)
If the return value is a *boolean* or an *integer*		
+12	4	numeric value (in network byte order)
If the return value is a *string*		
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)
If the return value is a *binary string*		
+5	2	string length nnn (in network byte order)
+7	nnn	string value (not nullterminated)
If the return value is a *real*		
		A UTF-8 string formatted as: [+|-]nnn.nnnnnn[e[+|-]nnn]
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)

Offset	Size	Description
If the return value is a date		
		A UTF-8 string formatted as: YYYYMMDD
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)
If the return value is a *time*		
		A UTF-8 string formatted as: `[YYYYMMDDT]hhmmss[uuu]` uuu is milliseconds The 'T' separates the optional YYYYMMDD date from the time value Note: this format is subject to difficulties between the client time zone and the MAES time zone
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)
If the return value is a *timespan*		
		A UTF-8 string formatted as: `#daysThhmmss[uuu]` uuu is milliseconds The 'T' separates the optional #days from the time value
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)
If the return value is a *isotime*		
		A UTF-8 string formatted as: `YYYYMMSSThhmmss[uuu][(+|-)GMTdelta]` uuu is milliseconds The 'T' separates the optional YYYYMMDD date from the time value The GMTdelta is the difference from Greenwich: example, -0500 indicates the time is 5 hours behind Greenwich

Offset	Size	Description
+12	4	string length nnn (in network byte order)
+16	nnn	string value (null terminated)
If the return value is a *timeval*		
		Note: this format is subject to difficulties between the client time zone and the MAES time zone.
+12	4	numeric time value, that is, A C "time T". (in network byte order)

Note: Subsequent output parameters follow immediately afterwards without any alignment adjustments. Output parameters are formatted identically to the return values above.

Java Resource Adapter (JRA) /Java Connector Architecture (JCA)

MAES Java EE Resource Adapter (JRA), and Java Connector Architecture (JCA) capabilities provide gateways to Aion BRE applications (KBs) that operate in a MAES TCP/IP-driven environment. These gateways are collectively termed JRA/JCA. The adapter programs should use platform neutral TCP/IP message communications as described in the previous section.

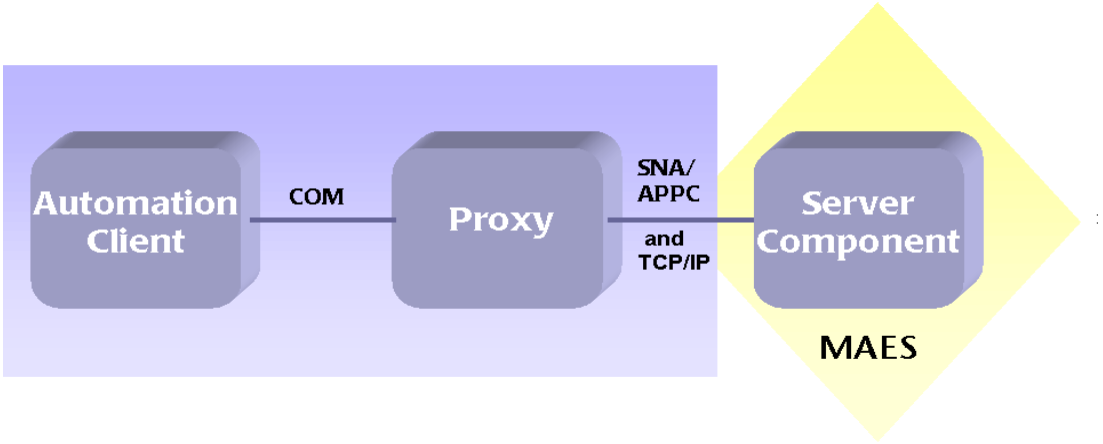
Create and Use Aion Components on PC Clients

This section discusses how to generate Aion components in which the client application runs on a PC and communicates with an Aion server component running on the mainframe.

CA Aion BRE supports PC clients using APPC or TCP/IP to access mainframe components. To build a component for deployment in MAES, use CA Aion BRE to generate a server and proxy (and optionally the client). These paragraphs describe the process:

- The server application is typically written on the PC, and then sent to the mainframe using FTP. This server application is compiled and executed on the mainframe under MAES, and is then used to generate a DLL file. The resulting DLL is a proxy implemented as a COM object. This proxy communicates to the mainframe server application using SNA or TCP/IP, and to the client application using COM.
- The client application is built on the PC, and accesses the server component through the generated COM proxy. The client application can be written in Aion, Visual Basic, or any other COM-supporting language.

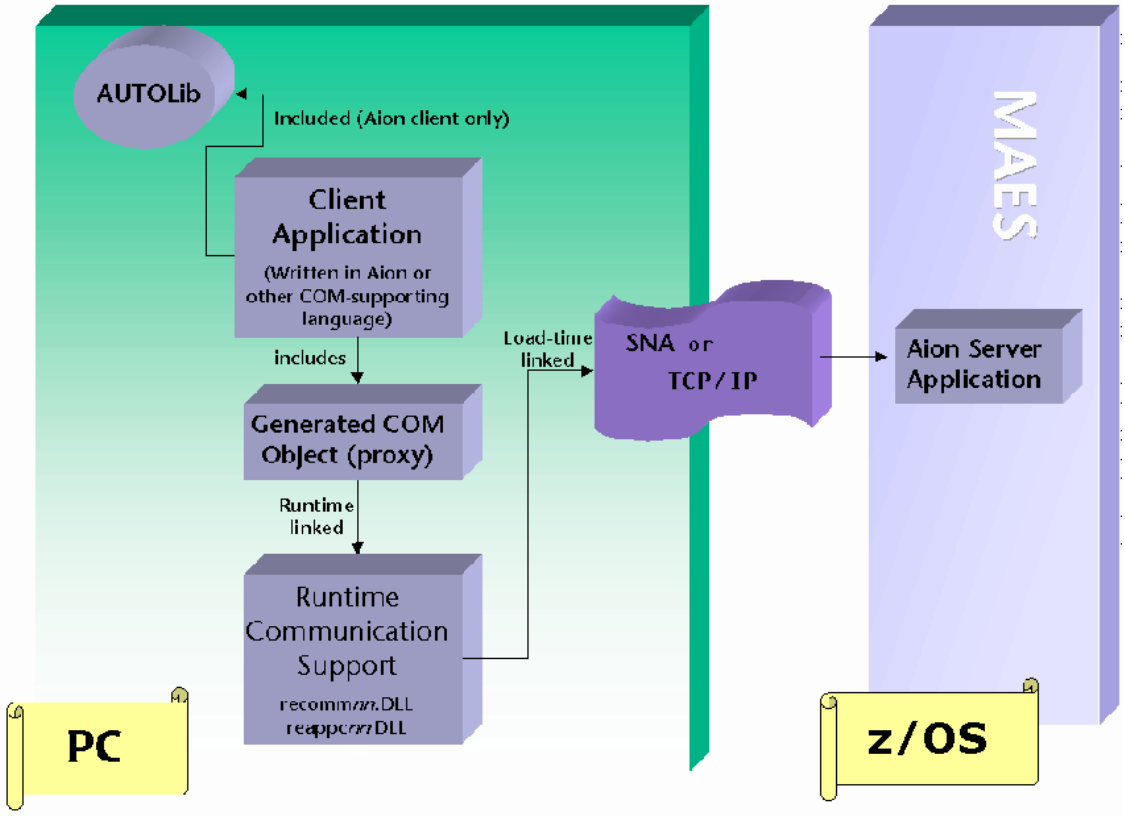
The following figure shows how the client uses a proxy component to connect to the server component:



The Windows-based client uses the Microsoft COM interface. For each Aion mainframe application, an z/OS COM component (proxy) is generated which acts as a COM object. This COM object exports all classes and methods contained in the server component.

Aion client applications include AUTOLIB, which is the Aion COM support library. The applications can then include the generated proxy DLL as a COM object and use its classes and methods.

The following illustrates a distributed Aion application:



Internally, the client proxy DLL is linked with the Aion communications support library, RECCOM*nn*.lib (which is automatically loaded at run time). When the APPC connection is established, RECCOM*nn*.DLL loads the appropriate APPC or TCP/IP support DLLs to establish the connection to the server.

Note: For r11.0 the module suffix (nn) is B0.

Generate a Proxy Component

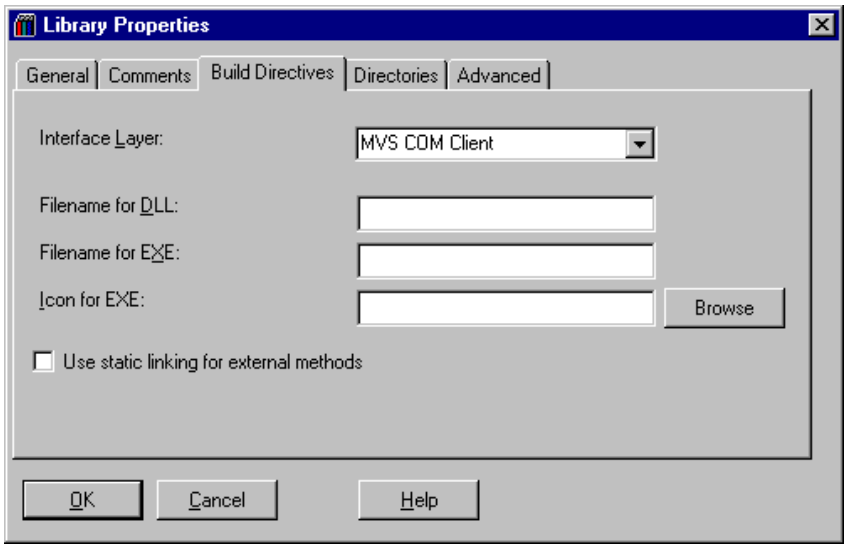
The proxy component is always generated on the PC from the Aion server application.

To generate a proxy component

1. On the PC, specify the Z/OS COM interface layer for the Aion server application:

   a. From the Aion IDE, highlight the server application's Libraries node (in the Project Workspace).

   b. Right-click and choose Properties from the pop-up menu.On the Build Directives tab, choose the Z/OS COM Client interface layer as shown in the following panel.



   c. Optionally, you can specify a custom name for the DLL. By default, the generated .dll is named *appname*.dll (where *appname* is the name of the server application). Click OK when finished.

2. Build the Aion server application to generate a DLL file and a TBL file.

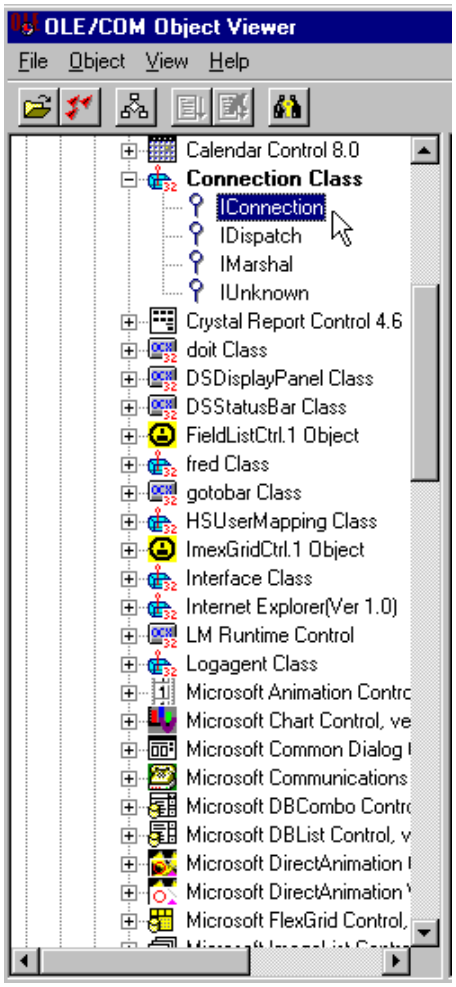
   The resulting DLL is a proxy implemented as a COM object that can communicate to the mainframe server application using SNA or TCP/IP, and to the client application using COM. In addition to exported classes from the server application, the proxy contains an automatically generated interface named IConnection. This interface contains the doconnect method, which supplies information the client uses at run time to locate and communicate with the server component.

3. Copy the built proxy DLL to the PC directory containing the client application.

   Note: If the client application is written in Aion, this directory should also contain the necessary Aion .dlls. By default, Aion DLLs are located in the same directory as redev.exe.

4. Register the proxy DLL on the PC (use regsvr32.exe).
5. Optionally, use OLEView to verify the object was registered (OLEView is supplied with Microsoft Visual C++).

   The OLE/COM Object Viewer is shown in the following figure:



6. **Note:** IUnknown and IDispatch are common to all COM server objects. The IConnection class is generated only when the Z/OS COM interface layer is used to build an application.
7. Next, generate the server component using the instructions provided in the next topic.

More Information:

Describe Your Environment Using the doconnect Method (see page 166)

Generate a Server Component

To generate a server component

1. Send the server application (used to create the proxy) to the mainframe using FTP.
2. On the mainframe, build the server application with a C interface layer.
3. Include the resulting DLL in the MAES STEPLIB concatenation.

More Information:

Build and Manage Aion Applications (see page 71)

Generate a Client Component

To generate a client component

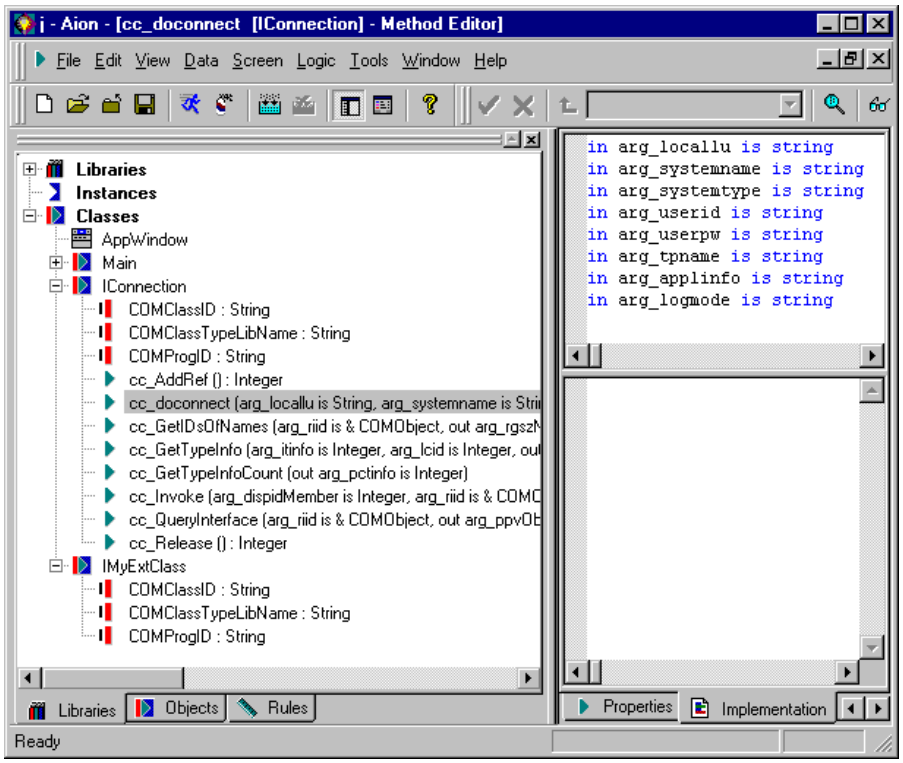
1. From the client application, call the IConnection doconnect method before other remote method calls.

   Here is an example in Visual Basic:

```
Dim x As Object
. . .
Set x = CreateObject("mydll.IConnection.1")
x.doconnect
```

2. Specify argument values for the doconnect method.

   The following figure shows the argument list for the Aion-generated doconnect method.



To Create an Aion Client

1. Open the client application in Aion, and include COM classes from the proxy (the copied and registered .dll file) as follows:

   a. Include AUTOLIB in your Aion application, and then from the Logic menu, choose New, COM Object.

   b. Click Browse to locate the .dll or .tbl file. The Select COM Objects dialog is displayed showing all interfaces available in the proxy.

2. In addition to your application classes, choose the IConnection interface from the proxy, and click OK to create the new IConnection class.

3. On the Libraries tab of the Aion Project Workspace, double-click the new IConnection class, and generate the doconnect method from the Generate COM Method dialog. Aion names this generated method cc_doconnect.

4. Include a call to IConnection.cc_doconnect from within your client application, specifying arguments to reflect your server environment.

More Information:

Describe Your Environment Using the doconnect Method (see page 166)

Describe Your Environment Using the doconnect Method

When the COM proxy .dll is generated (using the Z/OS COM interface layer), CA Aion BRE automatically creates an IConnection interface, which contains the doconnect method arguments you have supplied in the following syntax:

```
doconnect(locallu,systemname,systemtype,userid,password,tpname,applinfo,modename)
```

Note: All arguments are of type STRING.

The following lists the arguments you can use with the doconnect method to describe your environment:

locallu

Specify the name of the local (Windows) APPC logical unit (APPC).

Not used for TCP/IP.

systemname

Specify the name of the partner (MAES, CICS, IMS) logical unit for APPC, or the IP address in any of the standard formats, e.g. *xxx*.com, or 123.456.78.9 for TCP/IP.

systemtype

Specify for APPC, MAES, CICS, BAES (for IMS).

For TCP/IP, specify **TCP/IP.**

userid

Specify the user ID (if security is in place). Specify **""** if security is not used. This argument is optional.

password

Specify the password that authorizes the user (if *userid* was specified). Specify **""** if security is not used. This argument is optional.

tpname

For CICS, specify **AXFR.**

For IMS, specify **BAES.**

For MAES, specify **MAES.**

Not used for TCP/IP.

applinfo

Value must be **OAES.**

modename

Specify the Aion log mode name for APPC, **AIONMODE.**

For TCP/IP, specify the IP port number of the server (the same as that specified on the IPPORT parameter in MAES).

Sample Call to doconnect

The code that follows shows a sample call to doconnect:

```
var pConnection is pointer to IConnection
var handle is integer
var result is integer
// set up the connection
pConnection = IConnection.Create
if (pConnection = NULL ) then
    MessageBox(“cannot create connection”)
    return
end
//for APPC
result = pConnection.cc_doconnect(“RWC0050”,”OAES”,”SYSTYPE”,””,””,”OAES”,”EXE
OAES /TRAP(OFF)”,”AIONMODE”)
//for TCP/IP
result = pConnection.cc_doconnect(“”,”xxx.com”,TCP/IP”,””,””,””,”OAES”,”102”)
// start app window
```

Configure MAES for the Automation Server

To configure MAES for the Automation server, you must supply:

- The MAES region in which the Aion application DLL is available
- An LU name
- A VTAM definition of the LU name and node

To configure MAES as a TCP/IP server, you must specify the IPSTACK, IPTASKS and IPPORT run time parameters.

Use the following steps to build components that are accessed in MAES by a TP Monitor client application.

Test a Server Component on the PC

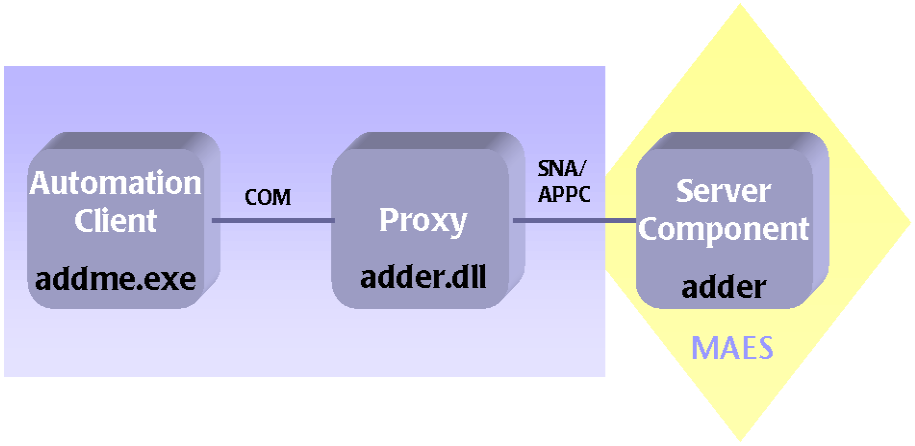
To test a mainframe server component on the PC, you build the application using a COM interface layer. You then register the proxy .dll and run the application on the PC to verify the application setup. When you are ready to move the server component to the mainframe, rebuild the application with the mainframe COM interface layer, and proceed as previously instructed.

Example

This distributed configuration example demonstrates an Z/OS COM client/server application in which the client is written in Aion. In this example:

- The built Aion server component is <*aionhlq*>.LOAD(ADDER), which identifies the fully qualified name of the Aion load library.
- The generated proxy is named adder.dll.
- The client application is named addme.app.

The following figure illustrates this example of a distributed configuration.



The topics that follow use this example to illustrate the procedures.

Build the Proxy Service

To build the proxy service

1. Using the z/OS COM client interface layer on the PC, configure the server application adder.app to be a COM server.
2. Build adder.app to create adder.dll and adder.tbl.
3. On the PC, the generated adder.dll serves as the proxy for communication between the client and the server.
4. Copy this adder.dll to the same location as the client application (addme.app).

Build the Server Component

To build the server component

1. Using FTP, send adder.app to the host.
2. Build the .APP file using the WRAPPER=C build directive, and then include the DLL in the MAES STEPLIB DD concatenation.

Register the COM Proxy

Register the COM proxy on the PC by typing:

```
regsvr32 adder.dll
```

You can use OLEView to verify the successful registration of the COM proxy (the COM classes defined by the proxy should appear under Automation Objects). The object node includes the automatically generated IConnection interface.

Configure the Client Application

The Aion client addme.app can now be modified to include methods described by the COM proxy, although their implementation actually exists on the mainframe. The client uses the IConnection interface to specify how the COM proxy should communicate with the mainframe, the MAES region, and the application under MAES.

To configure the client application

Using the Aion BRE IDE on Windows:

1. Import the COM interface IConnection:From the Logic menu, choose New, Com Object.

   Note: This menu option is available when AUTOLIB is included in your application.

   a. Click the Browse button and locate the proxy (adder.dll or adder.tbl).

   b. Highlight IConnection and click OK.

   The IConnection class is created in the application.

2. Generate the cc_doconnect method:

   Double-click the IConnection Class in the Aion Libraries tab, or right-click and choose Open from the pop-up menu. A list of methods in the IConnection class is displayed in the Generate COM Method dialog.Highlight the doconnect method and click Generate.

   The cc_doconnect method is created in the new IConnection class.

3. Use the cc_doconnect method:

   a. In ADDME.APP, call the IConnection.cc_doconnect method prior to referencing objects in the remote server component.

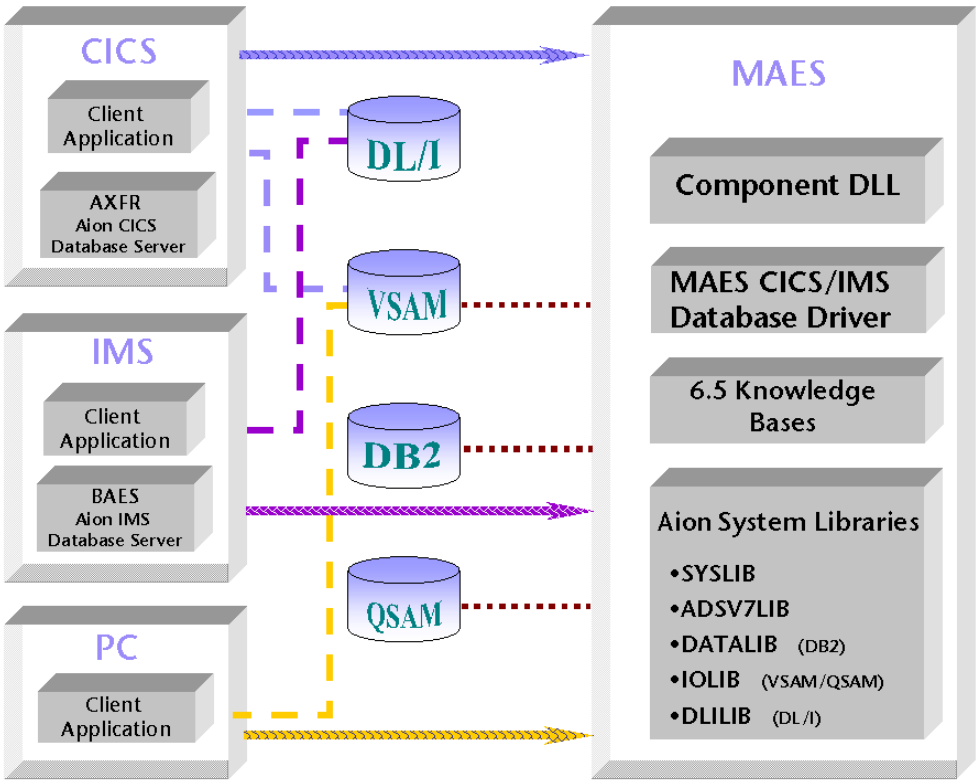
Chapter 5: The Multitasking Aion Execution System

The CA Aion BRE Multitasking Aion Execution System (MAES) permits access by multiple users to a single Aion application in a shared address space. This chapter describes how to use MAES, the MAES monitor, and MAES parameters.

This section contains the following topics:

The MAES Environment

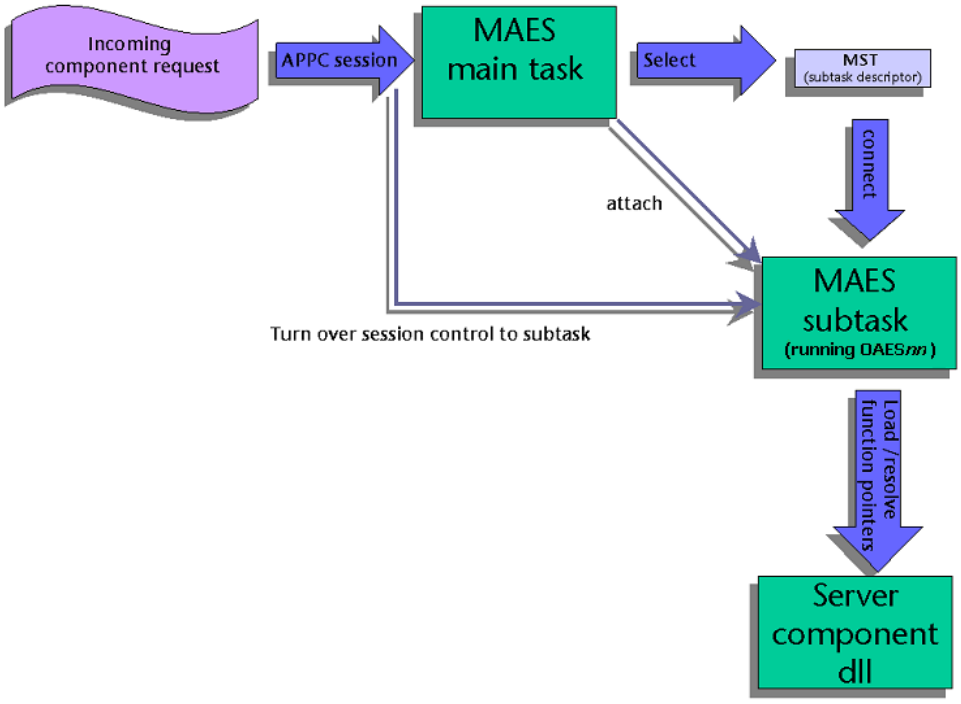
The following illustrates the MAES deployment environment:



MAES provides the environment for running components on a mainframe server. When a client establishes an APPC/VTAM session to MAES, the following events occur:

- MAES starts a new subtask running the CA Aion server component. This subtask is connected to a subtask descriptor (MST) chosen from the list of available descriptors.
- The client program loads the CA Aion interface DLL and issues function calls to the server component.
- As soon as the server component is initialized, it takes control of the APPC session, freeing MAES for other requests.
- When the server component terminates, it frees all its resources and returns control to MAES.

The following illustrates MAES request processing:



Start MAES

MAES runs as a job within the operating system. This job can be a started task, or it can be submitted as a batch job.

Execute the MAES Startup JCL

The MAES PROC is used to start the MAES region. Start MAES by submitting the installed MAES PROC JCL to z/OS. CA Aion BRE provides the sample JCL that you can use to start MAES. This JCL is supplied in the MAES member of the Aion PROCLIB.

To ensure adequate performance, set the MAES parameters to be equal to those of the CICS/VS or IMS/VS region with which MAES is communicating, and make MAES non-swappable. To customize your MAES startup JCL, use the instructions that follow this sample.

Note: MAES PROC resides in <aionhlq>.PROCLIB and the JCL to execute this PROC resides in <aionhlq>.MAES.JCL(MAESJOB).

```
//* MODIFY THE FOLLOWING PARAMETERS.
//*   AIONHLQ : HIGH-LEVEL QUALIFIER FOR AION SOFTWARE
//*   USERHLQ : HIGH-LEVEL QUALIFIER FOR USER APPS
//*   MAESHLQ : HIGH-LEVEL QUALIFIER FOR MAES SOFTWARE
//*   DB2HLQ  : HIGH-LEVEL QUALIFIER FOR DB2 SOFTWARE
//*   DB2XTHLQ: HIGH-LEVEL QUALIFIER FOR DB2 EXITS
//*   IBMHLQ  : HIGH-LEVEL QUALIFIER FOR IBM z/OS
//*             LANGUAGE ENVIRONMENT RUNTIMES LIBRARIES
//*   PARMS   : MEMBER OF MAESPARM CONTAINING PARMS
//*   LUNAMES : MEMBER OF MAESPARM CONTAINING LUNAMES
//*************************************************************
//*
//MAES  PROC AIONHLQ='SYS2.PROD',   <-- HLQ FOR AION BASE SOFTWARE
//           USERHLQ='USER.APPS',   <-- HLQ FOR USER APPS
//           MAESHLQ='SYS2.MAES',   <-- HLQ FOR MAES SOFTWARE
//           DB2HLQ='DB2.DB2810.GA',  <-- HLQ FOR DB2 SOFTWARE
//           DB2XTHLQ='D81B.PRIVATE', <-- HLQ FOR DB2 EXITS
//           IBMHLQ='CEE',         <-- HLQ FOR LANGUAGE ENV
//           PARMS=DEFPARMS,      <-- MEMBER IN MAESPARM FOR PARMS
//           LUNAMES=DEFLUNMS,   <-- MEMBER IN MAESPARM FOR LUS
//           PRELOAD=DEFPRELD,   <-- MEMBER IN MAESPARM FOR PRELOAD
//           LEPARMS=DEFLEPRM    <-- MEMBER IN MAESPARM FOR LE PARMS
//*
//MAES   EXEC PGM=MAES,REGION=0M,TIME=NOLIMIT,
//            PARM=(PMEMBER#&PARMS)
//STEPLIB  DD DSN=&MAESHLQ..LOAD,DISP=SHR
//         DD DSN=&AIONHLQ..LOAD,DISP=SHR
//         DD DSN=&IBMHLQ..SCEERUN,DISP=SHR     <-- z/OS LANGUAGE ENV
//         DD DSN=&DB2XTHLQ..SDSNEXIT,DISP=SHR  <-- DB2 EXITS
//         DD DSN=&DB2HLQ..SDSNLOAD,DISP=SHR    <-- DB2
//EDCMTF   DD DSN=&MAESHLQ..LOAD,DISP=SHR
//MAESHPO  DD DSN=&USERHLQ..MAES.LOAD,DISP=SHR
//         DD DSN=&USERHLQ..LOAD,DISP=SHR <-- COMPILED AION COMPONENTS
//ADSLIB   DD DSN=&MAESHLQ..LOAD,DISP=SHR
//MAESPARM DD DSN=&MAESHLQ..MAESPARM,DISP=SHR
//MAESPARM DD DSN=&MAESHLQ..MAESPARM,DISP=SHR
//MAESLUNM DD DSN=&MAESHLQ..MAESPARM(&LUNAMES),DISP=SHR
//COMTBLG  DD DSN=&MAESHLQ..COMTBLG(ENGLISH),DISP=SHR
//MAESPRLD DD DSN=&MAESHLQ..MAESPARM(&PRELOAD),DISP=SHR
//MAESAPRM DD DSN=&MAESHLQ..MAESPARM(&LEPARMS),DISP=SHR
//MAESLOG  DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSERR   DD SYSOUT=*
//SESSLOG  DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//CEEDUMP  DD SYSOUT=*
//ABNLIGNR DD DUMMY    <-- TO DISABLE LEAID
//CEEOUT   DD SYSOUT=*
```

Modify the Startup JCL PROC Statement

Customize the PROC section of the MAES startup JCL to specify your default values for the following parameters:

AIONHLQ

High-level qualifier for the CA Aion BRE product

USERHLQ

High-level qualifier for user applications

MAESHLQ

High-level qualifier for MAES components

IBMHLQ

High-level qualifier for the IBM z/OS language environment runtime libraries

PARMS

Name of the MAESPARM member containing parameters

LUNAMES

Name of the MAESPARM member containing LU names

PRELOAD

Name of the MAESPARM member containing Aion applications and modules to be preloaded

LEPARMS

Name of the MAESPARM member containing LE parms for Aion applications

DB2HLQ

High-level qualifier for DB2

DB2XTHLQ

High-level qualifier for DB2 exits

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

Modify the Startup JCL EXEC Statement

In the EXEC statement, specify the size of the MAES region and any MAES runtime parameters.

The following shows an EXEC statement in the MAES PROC:

```
//MAES    EXEC PGM=MAES,REGION=regionsize,
//             PARM=parameters
```

regionsize

Specifies the region size. This size must be large enough to accommodate the application you will run, as well as its data.

parameters

Specifies the MAES runtime parameters.

Note: If you are defining only a few parameters, you might want to specify them here. If you are defining many parameters, you may prefer to use the <aionhlq>.MAES.MAESPARM data set, which is identified during MAES installation.

Modify the DD Statements

For a list of DD statements and instructions for customizing the statements, see Specify MAES Runtime Parameters and DD Statements.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

MAES Generic Resource Support

CA Aion BRE supports online client access via VTAM Generic Resources (GR) to Aion components running in MAES. Generic Resource access to MAES is compatible with non-GR access to MAES by existing or new clients.

To access MAES using generic resources, MAES's GRNAME value is specified in the MAESNAME field for IMS clients or in the connection definition for CICS clients. For an explanation of the GRNAME parameter, see the section Specify MAES Runtime Parameters and DD Statements.

In order to access MAES components using generic resources, online client programs must use the ACQUIRE and RELEASE functions. See the sample client programs ending in '3' for examples'. Information regarding CICS client example programs is provided in <userhlq>.SAMPLES.README(CLIENT).

Pre-Initialized MAES Components

CA Aion BRE supports pre-initialization of CA Aion BRE components running in MAES. Pre-initialization of a component is similar to preloading, in that the Aion component DLL gets preloaded. However, with pre-initialization, the OAES subtask which services the CA Aion BRE component is loaded into memory and initialized. MAES can be configured to start as many copies of each Aion component as desired via entries in the MAESPRLD dataset.

In order to access pre-initialized MAES components, online client programs must use the ACQUIRE and RELEASE functions. See the sample client programs ending in '3' for examples. Information regarding MAES components example programs is provided in <userhlq>.SAMPLES.README(CLIENT).

More information:

Specify MAES Runtime Parameters (see page 193)

MAES Automation Facility

The MAES Automation Facility builds upon the MAES Generic Resource and Pre-Initialized Component capabilities and provides easier client coding and faster, more consistent client response time. To utilize the MAES Automation Facility, users must create a special class in the Aion application named MAES_Automation and marked for Export. In this class, users must create a public instance method named Init.

When the application is loaded, regardless of whether it is pre-initialized or dynamically loaded, MAES automatically invokes the Init method. The Init method can perform any initialization processing desired by the application, such as performing one-time DB2 access, setting up rules, and so on. The only coding requirement for the Init method is that it must provide an instance handle value which can be returned to client programs issuing Acquire calls. Since the instance handle is returned on the Acquire call, the client program no longer is required to make an additional call to create the instance handle required to execute the methods within the Aion application.

As an additional benefit, by creating an instance of the MAES_Automation class prior to executing the Init method, MAES automatically anchors the application symbol table. This ensures that the expense of creating the symbol table is only incurred once. For pre-initialized applications, the creation of the symbol table occurs prior to access by a client program, thus reducing client response time and making it more consistent. The application must not delete the MAES_Automation class instance because MAES saves its value and uses it later.

Users can also create a public instance method named Release in the MAES_Automation class. When the client program releases the Aion application, MAES automatically invokes the Release method. The Release method can perform any processing required to prepare the application for reuse by the next caller. The only coding requirement for the Release method is that it must provide an instance handle value which can be returned to client programs issuing Acquire calls. The instance handle returned by the Release method can be the same as the instance handle returned by the Init method, if desired. However, some applications may find it easier to delete all the instances of the application classes and simply create a new accessor instance.

As an additional benefit, the Release method of the MAES_Automation class is executed asynchronously. More specifically, control is returned to the client program immediately upon receipt of a release request and MAES executes the Release method while the client program continues to process. This reduces client response time by the amount of time required to prepare the Aion application program for reuse. MAES uses the value of the MAES_Automation class instance created at application initialization time to access the Release method repeatedly. This is why the application must not delete the MAES_Automation class instance.

While it is to the user's advantage to code a Release method in the MAES Automation class, it is not required. However, if a Release method is not coded, some other method called by the client program must prepare the Aion application for reuse and the client program response time will be increased by the amount of time required to prepare the Aion application for reuse. Also, without a Release method, the instance handle returned by the Init method cannot be altered, precluding the option of deleting and recreating the accessor instance.

Both the Init and Release methods return an integer value. Normally, this return value should be zero (0). However, if an error condition is encountered by the Aion application, a non-zero value can be returned and the Aion application terminated by MAES. For pre-initialized Aion applications, each copy of the application is given an opportunity to initialize. If a temporary problem (such as a database condition) causes an insufficient number of copies to initialize, additional copies are started automatically when Acquire requests arrive. If the Init method return value is non-zero for a dynamically started Aion application, the client program receives an error code indicating that the Aion application was terminated. Regardless of when Aion applications are terminated due to non-zero Init or Release return values, if the supply of available Aion applications is exhausted, MAES attempts to start a new copy dynamically when the next Acquire call is received.

In order to utilize the MAES Automation Facility, online client programs must use the ACQUIRE and RELEASE functions. See the Objserv sample Aion application for an example of how to code the Init and Release methods in the MAES_Automation class. See the sample client programs ending in '4' for examples of client programs using the MAES Automation Facility.

XPLINK MAES Components

CA Aion BRE fully supports XPLINK Aion components running in MAES. Each XPLINK component executing in MAES must have an entry in the MAESAPRM dataset identifying it as an XPLINK component.

MAES XPLINK components may be accessed by any online client programs.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

Suspend/Resume MAES Components ('Hot' Apps)

CA Aion BRE supports Suspend/Resume functionality for Aion components running in MAES. This allows the client program to suspend a component in MAES and free the APPC session, and then resume that component's activity at a later time using a new APPC session and continue processing. MAES keeps the component server task in a suspended state and keeps the component DLL loaded. All memory content remains available.

The Suspend/Resume functionality allows online client programs to access reusable ("hot") apps executing in MAES. There are two options for implementing *hot* apps:

- Client-managed (using RESUME and SUSPEND)
- MAES-managed (using ACQUIRE and RELEASE)

While the performance of these two options is comparable, the MAES-managed option offers several advantages.

With client-managed *hot* apps, the online client is responsible for managing the *hot* apps. This requires a special client program (commonly a CICS PLT startup task) to start up whatever number of *hot* apps is desired (assuming a different task ID for each copy), SUSPEND each copy of the app, and save off the associated task IDs in a database table. Later, client programs access the database table to locate an available *hot* app task ID, update the table to reserve the app, assume the task ID of the available app, RESUME the app, use the app as desired, SUSPEND the app, and finally update the table to indicate that the app was once again available for reuse.Refer to <aionhlq>.SAMPLES.README(CLIENT) for example client programs, see the sample client programs ending in 2 .

With MAES-managed *hot* apps, MAES automatically initializes the desired number of *hot* apps and will dynamically start additional *hot* apps should the need arise. All the client program has to do is ACQUIRE the app, use the app as desired, and RELEASE the app. To have MAES automatically initialize *hot* apps, see the MAESPRLD DD statement in the section Specify MAES Runtime Parameters and DD Statements. Refer to <aionhlq>.SAMPLES.README(CLIENT) for example client programs, see the sample client programs ending in 3.

Important! CICS client programs using the RESUME and SUSPEND or ACQUIRE and RELEASE functions must ensure that the APPC conversation ID (EIBRSRCE, an 8-byte character field in the EIB), remains constant for all function calls between the RESUME and SUSPEND or ACQUIRE and RELEASE calls. The EIBRSRCE field contains the ID of the APPC conversation with MAES and is overridden by CICS with the conversation ID for the user's terminal. Unless a client program makes all its component calls between terminal SENDs, the EIBRSRCE value must be saved prior to each terminal I/O and restored afterwards. Otherwise, the next method call fails because the connection to MAES cannot be identified correctly.

Run the MAES Monitor

Use the MAES monitor to perform the following tasks:

- Display active processes and their statistics.
- Display historical data for a given MAES identifier.
- Display the current MAES version.
- Perform maintenance functions on session partners.
- Terminate a process.
- Print an internal activity trace showing the VTAM activity performed within the MAES environment.
- Record the time taken for various execution tasks during a specified interval.
- Display the XA and non-XA memory currently in use by all system processes.
- Terminate the MAES address space.

Run the MAES Monitor from TSO

You can use the TSO monitor utility MAESMON to connect applications in MAES to the MAES monitor from TSO. This utility can be invoked by a simple CLIST. A sample CLIST Is provided in the <aionhlq>.CLIST as member MAESMON. When you invoke MAESMON, a dialog is presented that allows you to type the MAES LU name, the local LU name, and command line options.

```
MAES LU name  . . MAES

Local LU name . . AION
Cmd line options


F1=Help     F3=Exit    F12=Cancel
```

The MAES Monitor

When you activate the MAES monitor, the MAES Monitor Main Menu is displayed, as shown in the following example:

```
Aion MAES Monitor                                4-May-2009  1:02pm v11.0

                                Monitor Main Menu

                            Enter S next to your selection

                          -  ------------------------------
                          S  Display Current Consultations
                             Display Historical Information
                             Display Product Versions
                             Manage Optimized v6.5 Programs
                             Manage Session Partners
                             Cancel a Consultation
                             Print VTAM Activity Trace
                             Manage Timing Analysis
                             Monitor Storage Utilization
                             Kill MAES

                             Quit
                          -  ------------------------------

 F1=Help    F3=Cancel    F12=Exit Monitor
```

The following topics describe the tasks that can be performed from this menu.

More Information:

MAES Reports: Gathering Statistics (see page 225)

Display Current Consultations

Choose the Display Current Consultations option to examine the internal MAES control blocks to see which subtasks are running. The subtasks can be active or suspended (waiting for user input). Your current MAES monitor consultation is always shown as active. You can choose Quit to return to the MAES Monitor Main Menu.

The following example shows the active Consultation Display panel:

```
Aion MAES Monitor                                      4-May-2009  1:02pm v11.0

 ------------------------------------------------------------------------------
                              Aion MAES Monitor
                            Consultation Display


 Idnt      Kbname   Userid    Ltname   Himem   Maxmem  # Actions
 ----     -------- --------  -------- ------- -------- ---------
 A001     MONITOR  CICS410T  1129     3527952  4096000       2

     Start time :  1:02pm CPU used :  0.121 (*** Active ***)

                         In              Out
                         -------         -------
     VTAM Rus            7               6
     Screen              2               2
     Data                0               0
     Other               5               4
 ---------------------------------- More >----------------------------------

 F1=Help    F3=Cancel    F12=Exit Monitor
```

Display Historical Information

To display the cumulative activity of all completed subtasks, choose the Display Historical Information option. Choose Quit to return to the MAES Monitor Main Menu.

Note: Unless you enable the MAES accounting function, historical information is not captured and is not available to the MAES monitor.

The following example shows the Historical Consultation Data panel:

```
Aion MAES Monitor                                  4-May-2009  1:02pm v11.0

 ------------------------------------------------------------------------------
                              Aion MAES Monitor
                          Historical Consultation Data
Identifier: A001
 Number of consultations: 2

        Average consultation CPU time :  0.000

                         In              Out
                         -------         -------
     VTAM Rus            41              39
     Screen              13              13
     Data                0               0
     Other               28              26
 ---------------------------------- More >----------------------------------

 F1=Help    F3=Cancel    F12=Exit Monitor
```

More Information:

Accounting Record Format (see page 255)

Display the MAES Product Version

Choose the Display Product Version option to display the current version and release of MAES. Choose Quit to return to the MAES Monitor Main Menu.

Manage Session Partners

You can manage the current set of session partners to optimize security and performance. When you choose the Manage Session Partners option, the Manage Session Partners sub-menu is displayed. You can perform one of the following tasks without having to restart MAES, or you can choose Quit to return to the MAES Monitor Main Menu:

- Display the current session partners.
- Activate a session partner.
- Deactivate a session partner.
- Add a new session partner.
- Acquire a session partner.

Display Session Partners

Choose this option to identify all current session partners and include the following information for each:

- VTAM LU name
- Type of session partner (CICS, IMS, or independent)
- Whether the session partner is active
- Whether the session partner is bound to MAES
- The VTAM Log mode table entry name

Activate a Session Partner

Choose this option to make a session partner available for binding and, therefore, available for further consultation requests. Activating a partner does not rebind the VTAM session. Activating a partner simply makes that partner available for binding.

Deactivate a Session Partner

Choosing this option causes the following two events to occur:

- All bound sessions with that partner are released.
- The session partner is flagged as being unavailable for any bind requests.

  The deactivated session partner is disabled for any further consultations.

Note: To prevent potential abends, do not deactivate the session used by the MAES Monitor.

Add a New Session Partner

Choose this option to dynamically add a new session partner. The definitions of the new session partner do not change in the MAESLUNM data set. This option causes MAES to add new session partners to the existing definitions controlled by MAES. The additions are lost when MAES terminates.

Acquire a Session Partner

Choose this option to rebind with a CICS/VS session partner if you did not automatically acquire it. This option also allows you to reacquire sessions with CICS/VS.

Cancel a Consultation

Choose the Cancel a Consultation option to cancel a subtask that is looping or otherwise not responding.

Print VTAM Activity Trace

MAES maintains an internal trace of message activity on VTAM sessions. Choosing the Print VTAM Activity Trace option sends the trace, in report form, to the data set associated with the SESSLOG DD statement in the MAES JCL, as in this example:

```
//SESSLOG DD SYSOUT=*
```

To write multiple trace outputs to an existing data set, allocate the SESSLOG DD name as shown:

```
//SESSLOG DD DSN=SYS2.VTAMTRAC.OUTPUT,DISP=MOD
```

More Information:

Accounting Record Format (see page 255)

Manage Timing Analysis

Choose the Manage Timing Analysis option to activate or deactivate the MAES Timing Analysis function. Activating MAES Timing Analysis causes records to be written to the data set associated with the MAESMTAR DD statement in the MAES JCL.

More Information:

Perform MAES Timing Analysis (see page 235)

Monitor Storage Utilization

Choose the Monitor Storage Utilization option to display the amount of XA and non-XA memory currently being used by all subtasks. The z/OS VMSLIST macro (SP=PVT) is issued for each subtask under MAES to determine the private storage usage. SP=PVT obtains allocated storage information for subpools zero through 128, 229, 230, 236,237,251, and 252.

Terminate MAES

Choose the Kill MAES option to terminate MAES. When you terminate MAES, the MAES monitor subtask ends abnormally. This option lets you simulate the effects of the console operator replying **t** to the outstanding WTOR (Write to Operator with Reply). Internally, the WTOR is answered, which causes MAES to end its VTAM connections. The transaction driver detects the loss of the VTAM session and treats it as an error condition.

Note: You can set the KILLMAES parameter to specify whether a user can terminate MAES from the MAES monitor application.

There are several methods you can use to terminate MAES.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)
Terminate MAES (see page 225)

Employ Authorization Exit Routines

This section discusses the following topics:

- Activating the authorization exit routines
- Coding the program interface
- Using Aion-supplied exit routines

Note: The authorization exit routines are different from external security routines, such as RACF and CA-ACF2, which MAES can access.

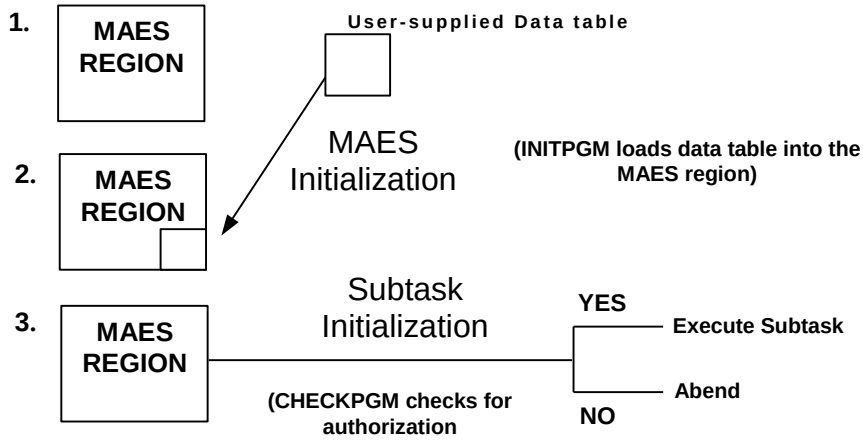
Activate the Authorization Exit Routines

An authorization exit can be placed at two points in MAES processing: during the MAES initialization, or inside each subtask initialization.

During MAES initialization, the authorization exit allows you to set up tables and other environmental data which the second authorization exit uses. For example, with the first authorization exit, you can load a list of authorized users and the programs they are authorized to run in the MAES region address space.

The second authorization exit can be placed in each subtask initialization. This authorization exit uses the information loaded by the first authorization exit to determine if the user is authorized to run the subtask.

The following illustrates an example of MAES initialization setting up a data table used by the subtask initialization:



Specify the executable program names in the MAES runtime parameter, PGMSEC. The syntax for the PGMSEC parameter is:

```
PGMSEC=(initpgm,checkpgm)
```

initpgm

Is the name of the user-specified authorization exit that runs when the first exit is taken during MAES initialization. It causes the data table or other environmental information to be loaded into the MAES region.

checkpgm

Is the name of the user-supplied authorization exit that runs when the second exit is taken during subtask initialization. It checks for user authorization.

Note: The name identified by *checkpgm* must be reentrant and reusable. Both authorization exits must be capable of 31-bit addressing. Additionally, both authorization exits must be accessible through the STEPLIB or JOBLIB DD statements in the MAES JCL.

If you do not have an initialization program, but instead have an authorization table assembled and link-edited as *checkpgm*, then use the following syntax for the PGMSEC parameter:

```
PGMSEC=(,checkpgm)
```

Code the Program Interface

Using a z/OS LINK request, MAES runs the initialization program you have specified by *initpgm*. The parameter list follows standard operating system linkage conventions, as presented in the following descriptions:

Register 1

Register 1 points to the following addresses:

- The fullword address of an internal control block. This is used by an Aion-supplied routine to write information to the MAES log.
- The fullword address of a fullword that can be modified by *intipgm*. This points to a fullword in which you can store an address, a value, or anything else *checkpgm* uses.

Register 15 (After initpgm)

Upon return from *initpgm*, register 15 contains one of the following values:

- A zero value indicates that the program ran without an error.
- A nonzero value indicates that an error occurred. After printing an error message, MAES terminates.

If register 15 contains a nonzero value, an error message is printed, and the subtask initialization program *checkpgm* is given the following parameter list, which follows standard operating system linkage conventions:

- The fullword address of the *initpgm* fullword
- The fullword address of an eight-character program name
- The fullword address of an eight-character user ID

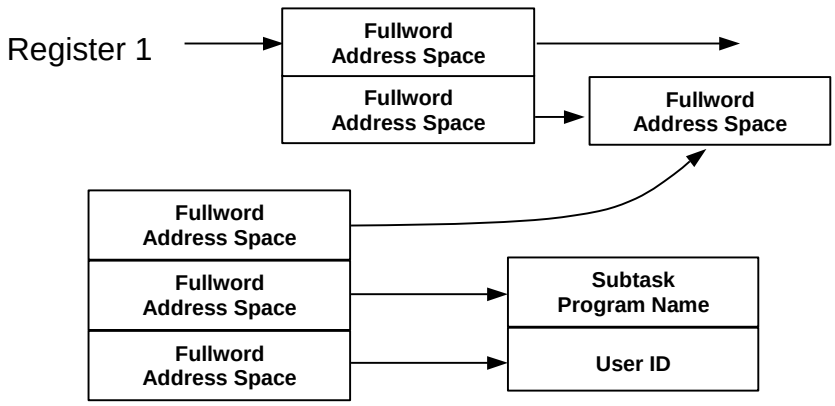
The program name and user ID are left-justified and filled with blanks.

Register 15 (After checkpgm)

Upon return from *checkpgm*, register 15 contains a value:

- A zero value indicates the user is authorized and the subtask is started.
- A non-zero value indicates that an error occurred. If register 15 contains a non-zero value, the MAES log contains a log message indicating that the authorization check failed, and the consultation abends with user abend 377.

The following figure shows the linkages that form the program interfaces for the security exits:



Supplied Exit Routines

Instead of writing your own code, you can use two exit routines supplied with CA Aion BRE. To use these initialization and checker programs, specify the following syntax for the PGMSEC runtime parameter in the MAES JCL PARM statement:

```
PGMSEC=(maespgsi,maespgsc)
```

maespgsi

Is the name of the initialization program.

maespgsc

Is the name of the checker program.

The MAESPSEC DD statement in the MAES JCL must be allocated to a sequential data set defined with 80-character records. Use the following syntax to identify the users, and the modules each user is authorized to run:

```
USER(userid1,userid2,...,useridn)    PGM(progname1,progname2,...,prognamen)
```

userid

Is the name of a user.

progname

Is the name of the module the user is authorized to run.

Each statement specifies one or more users and the module(s) each user is authorized to run. If you do not explicitly authorize any users to run a module, *no one* will be able to run the module. The modules must be located in a library specified in the MAESHPO, STEPLIB, or JOBLIB DD statements.

You can specify that all users or programs are authorized by coding an asterisk (*) in the USER or PGM statements. For example, all users are authorized to execute the program PGMA using the following statements:

```
USER(*) PGM(PGMA)
```

Note: You can use a hyphen (-) at the end of a record to continue the input statements.

Assume, for example, that you want to authorize all users to run a knowledge base, yet allow only USER1 and USER4 to create or edit a knowledge base. You also want USEROPER to run the MAES monitor knowledge base. You would specify the following input statements:

```
USER(*) PGM(SAES)
USER(USER1,…,USER4) PGM(SADS)
USER(USEROPER) PGM(MONITOR)
```

Specify MAES Runtime Parameters and DD Statements

Runtime parameters are specified in the MAES member of the <aionhlq>PROCLIB. Most of these parameters are specified during installation, although a subset can also be modified at runtime.

To specify MAES runtime parameters, use *one* of the following methods:

- The EXEC statement of the MAES JCL stream
- Source statements in a member of a partitioned data set associated with the MAESPARM DD

EXEC Statement

The following statement illustrates a MAES EXEC statement with parameters:

```
//MAES EXEC PGM=MAES,PARM='DEBUG=NO,MAESLOG=YES'
```

Enclose each line of parameters (after PARM=) in single quotes, and separate one parameter from the next using a comma.

Source Statements

To specify runtime parameters as source statements in a PDS member referenced by the MAESPARM DD statement, use the PMEMBER runtime parameter. PMEMBER is discussed in the next topic.

Specify MAES Runtime Parameters

The following paragraphs describe Aion MAES runtime parameters. The parameters are presented in alphabetical order.

CPUTIME

```
CPUTIME=cputime
```

You can specify the maximum CPU time allowed between terminal interactions before MAES abends or times out a consultation. The value is expressed in base 10 as a 1 to 8-character string of numeric digits, with a decimal point optional.

MAES starts the timer when it receives user input from the transaction driver. MAES stops the timer when the next output screen is sent to the transaction driver. If the timer runs out before MAES sends the next screen to the transaction driver, the consultation ends with a user abend 4000. The default is five (5) seconds. You can change the value using the MODIFY command.

BUSYPCTLVL

```
BUSYPCTLVL=nn
```

Health Checker uses this parameter to measure response threhold. The default is 20, which means a 20% busy response threshold.

DEBUGKB

```
DEBUGKB=[YES|NO]
```

MAESDBUG is to be called when errors occur during MAES processing. MAESDBUG helps you locate and correct errors that occur during the execution of MAES.

For example, if MAES is unable to establish a connection to VTAM, MAESDBUG interprets the error code and offers suggestions about how to resolve the problem. If the value of the MAESLOG runtime parameter is YES, the output of MAESDBUG is written to the MAESLOG data set. (We recommend that you route the information to SYSOUT rather than to a permanent data set.)

DIVLOGDS

```
DIVLOGDS=dataset.name
```

This parameter specifies the Z/OS data set name for the MAES log. This parameter is required. DIVLOGDS cannot be allocated to an SMS managed volume. Use DFSMS to force allocation to a non-SMS managed volume.

The data set name can be up to 37 characters, with a final qualifier of six characters or less. MAES adds a two-character numeric suffix to the final qualifier for dataset series and there is an implicit "DATA" that VSAM adds to the end of the dataset. For example:

```
SYS2.AION.MAES.MLOG
```

Becomes:

```
SYS2.AION.MAES.MLOG00
```

```
DIVLOGSZ=nnnn
```

This parameter specifies the size of the MAES log in 4K blocks. This parameter is required.

This number must be a multiple of 16. If you specify a number that is not a multiple of 16, MAES converts it internally (for example, 500 becomes 496).

One 4K block can contain 32 messages. For example, a DIVLOGSZ of 496 holds 15872 messages.

ESTAE

```
ESTAE=[YES|NO]
```

You can turn off the ESTAE and SPIE functions that are issued by MAES. This prevents MAES from doing a controlled shutdown if an abend occurs. It is most useful in debugging a program check. If ESTAE=YES (the default) is specified, the actual abend is masked, making debugging more difficult. If ESTAE=NO, then MAXABEND=1.

Note: ESTAE must equal YES when VTAM Generic Resource support is being used in order to ensure that the MAES shutdown routine gets control and is able to delete VTAM affinities. If ESTAE is set to NO, MAES's shutdown routine will be unable to get control during abnormal MAES terminations and VTAM will be unable to reroute connected GR clients to other MAES systems.

GRNAME

```
GRNAME=vtamgrid
```

This parameter allows the systems programmer to specify the name of a VTAM Generic Resource (GR) pool that MAES should participate in. The default value is MAESGR. You can override the name for the following reasons:

- Installation standards require a name other than MAESGR, which is the default.
- The systems programmer wants to support two or more MAES generic resource pools.

The value specified for *vtamgrid* cannot be longer than eight characters. Do not make an entry for *vtamgrid* in the VTAM tables. VTAM will dynamically provide any required definitions.

IPPORT

```
IPPORT=nn
```

This parameter identifies the TCP/IP port number that MAES uses to establish TCP/IP connections. In TCP/IP terms, MAES listens on this port.

IPTASKS

```
IPTASKS=nn
```

This parameter sets the number of concurrent TCP/IP components that can be started in the MAES region. If not specified, TCP/IP components cannot be run.

IPSTACK

```
IPSTACK=IPV4 | IPV6 | ANY
```

This parameter identifies whether MAES TCP/IP components will use IPV4 communications protocols only, IPV6 communications protocols only, or both IPV4 and IPV6 communications protocols. The default, when the IPSTACK parameter is omitted, is ANY.

ITRACE

```
ITRACE=nnn
```

ITRACE defines the number of internal trace entries MAES maintains per VTAM session. The internal trace is used by CA Aion BRE support for debugging purposes. The default is 10. A value of zero disables the internal trace.

KILLMAES

KILLMAES=[YES|NO]

This parameter determines whether a user running the MAES monitor can terminate MAES from there. If set to NO, the KILLMAES parameter takes no effect. The default is YES.

Note: Because Monitor is an application that runs in the MAES region, the MAES monitor subtask abends with a consultation error when you terminate MAES using the KILLMAES parameter. This method of terminating MAES can result in an A03 abend that could prevent other steps in the MAES job from running.

MAESLOG=[YES|NO|(*errn-errn*,…,*errn-errn*)|LOG_#S]

A low-level logging facility is built into MAES to aid in debugging message traffic. The output from this log is written to the MAESLOG data set. (We recommend that you route the information to SYSOUT rather than to a permanent data set.) This log is particularly useful when Aion Support is required to help solve a problem.

Since this log can produce a great deal of information, it is recommended that MAESLOG=YES (the default) be specified for non-production systems. However, should a problem arise in a production environment, this log can be of value while troubleshooting.

If you want to limit the range of error messages produced, you can define a range of log numbers for the MAES log to show when these specific errors occur. For example, the following command specifies that MAES write only the errors having the codes 1-1000 and 3000-4000 to the MAES log:

MAESLOG

MAESLOG=(1-1000,3000-4000)

To return only component messages, specify the following:

MAESLOG=(18000-19000)

If you wish to omit specific message numbers from the set to be printed, you must first specify:

MAESLOG=NO

Specify all the numbers you wish to be printed. Otherwise, all modifications add to the list.

You can change the value using the MODIFY command.

MAXABEND

MAXABEND=*nn*

MAES uses this value to determine the number of MAES abend conditions that can occur before MAES terminates.

Using MAXABEND, you can specify the number of MAES abend conditions that MAES should ignore before it terminates with an error. Consultation abends are not affected by MAXABEND. If the execution parameter ESTAE=NO, then MAXABEND=1.

The default is 10. The MODIFY command allows you to change the value.

MAXTASKS

MAXTASKS=*nn*

MAES uses the value you specify to control the number of concurrent consultations allowed at any given time.

This number includes not only active consultations, but suspended consultations. If a consultation start request is made after the maximum number of tasks is reached, the requester is given a status message and the consultation request is rejected.

The default is 10. MAXTASKS *cannot* be changed using the MODIFY command.

MTAR

MTAR=[YES|NO]

When the MAESMTAR DD statement is included in the MAES JCL, the MAES parameter can used to activate MAES Timing Analysis. Specify MTAR=YES as a MAES runtime parameter when you activate MAES.

NODENAME

NODENAME=*applid*

This parameter allows the systems programmer to override the default VTAM application identifier name (*applid*). You can override the name for the following reasons:

- Installation standards require a name other than MAES, which is the default.
- The systems programmer wants to run two or more copies of MAES.

The value specified for *applid* cannot be longer than eight characters, and must match the entry in the VTAM tables.

PGMSEC

```
PGMSEC=(initpgm,checkpgm)
```

The PGMSEC keyword is used to identify one program that is called during MAES initialization, and another that is called during knowledge base initialization. The intent of these two programs is to provide a level of security that can prevent certain users from executing one or more programs. The most obvious use would be to restrict use of the MAES monitor to a set of developers.

The *checkpgm* specification is mandatory; the *initpgm* specification is optional.

PMEMBER

```
PMEMBER=pds.member.name
```

You can specify execution parameters without using the EXEC statement or the MODIFY command. Do this by capturing the appropriate (desired) parameters as members of a PDS associated with the MAESPARM DD statement.

For example, you might have a set of MAES log specifications that you want to use for debugging at a certain time of day. At the same time, you want to change the wait time limit for a consultation to allow for longer periods. Assume you have coded the following entry and put it into a member named OVERNITE in the PDS associated with the MAESPARM DD statement:

```
MAESLOG=(1-3000,5001,6000-9000)
TIME=2000
```

At the appropriate time, the console operator types the following statement:

```
MODIFYjobname,PMEMBER=OVERNITE
```

Assuming the MAESPARM DD statement is present in the MAES JCL, the member OVERNITE is read and interpreted, and new values are set. If there is no MAESPARM DD statement, the request fails.

PMEMBER can be embedded within other members. Assume, for example, that you have a member named HOLIDAY, which is coded as follows:

```
CPUTIME=10.00
PMEMBER=OVERNITE
```

The CPUTIME is set to 10 seconds and the OVERNITE member is read and interpreted.

You can change the value of PMEMBER using the MODIFY command.

Note: MAES recursively processes all PMEMBERs during startup. In the case where the multiple PMEMBERS parameters are defined, during MAES startup, the parameter values will be set to the values contained in the last PMEMBER processed not the first.

ROUTCDE

```
ROUTCDE=nn
```

Certain actions within MAES produce operating system WRITE TO OPERATOR (WTO) requests. If, for example, the operator modifies the MAES job using the MODIFY command, completion information is presented as a WTO.

Using the ROUTCDE operand, you can direct the output to an appropriate destination. Values can be between 1 and 16, inclusive. ROUTCDE=11 writes the messages to the job log. This is commonly called a WRITE TO PROGRAMMER request.

The format of the operand is:

```
ROUTCDE=nn
```

or

```
ROUTCDE=(nn,nn,...,nn)
```

The default is 11. You can change the value using the MODIFY command. Consult your systems programmer for specific route codes used for your installation.

SEC

```
SEC=progname
```

The SEC parameter allows you to specify the name of a program that will be called every time a consultation is initiated. The user-written program must be re-entrant, and will be loaded at MAES initialization and then called by SAES or SADS (Subtasking Aion Development System).

The intention of this program is to allow a user-written security program to set the security environment so that subsequent data set allocations or OPENs will acquire the security authorizations of the user who is executing the consultation. The user-written program is passed to the user ID as it was retrieved by the ATD in CICS/VS, IMS/VS, or TSO. With this information, the user-written program can invoke the correct security function to establish that ID as the user-of-record for any dynamic data set allocations, or data set OPENs required by the knowledge base.

If remote DB2, remote VSAM, or DL/I is used exclusively, security is managed in the CICS/VS or IMS/VS environment. If nonremote VSAM, QSAM, or DB2 functions are used by the knowledge base, security needs to be established for the user.

Note: If security has not been established, the user ID will be blank.

STATINT

```
STATINT=hhmmss.th
```

The STATINT execution parameter sets the statistics-gathering time interval. The value is expressed as *hhmmss.th*, where:

hh

*i*s hours.

Mm

*i*s minutes.

Ss

is seconds.

.th

*i*s tenths and hundredths of a second.

A zero value disables the statistics gathering function. The default is 00003000. You can change the value using the MODIFY command.

TASKMEM

```
TASKMEM=nnnn[M|K]
```

TASKMEM sets the upper boundary on the amount of virtual memory a given application can acquire. This memory is *not* preallocated, and is outside of any user-coded program storage requirements.

The storage requirements of lists of items, records, and other information a knowledge base might specify, will be counted until the next request would exceed the limit specified by the TASKMEM parameter. When the limit is exceeded, the consultation is ended abnormally, and a status message is sent to the session partner.

The purpose of this parameter is to prevent a condition where a consultation will run today, but would fail tomorrow because of the requirements of other concurrent consultations. Use of this runtime parameter allows the systems programmer to better balance the requirements of different knowledge bases that are contending for storage in the same address space.

Approximately 10KB of storage is acquired for each consultation by operating system services. That 10KB is not counted when MAES is monitoring the storage requests.

Valid values for this keyword are *nnnn*, *nnnn*M, or *nnnn*K, where *nnnn* is a number. K indicates the number of 1024-byte blocks of memory available to the consultation, and M is the number of 1MB blocks of memory available. The default is 1M.

TIME

TIME=*nnnn*

MAES allows you to set a limit on the amount of time a process waits for the user to respond with the next input. The time is specified as the number of seconds SAES will wait between a message sent to its session partner and a message is returned to SAES. When this time is exhausted, the process is terminated and its resources are returned to MAES. Typically, this parameter would be used if the resources available to MAES are limited and you want to ensure that processes are not idle for extended periods.

If you wish to disable the wait time function altogether, you can specify:

TIME=0

If you specify TIME=0, MAES will not monitor process wait time, allowing a user to begin a process and then perform some other activity before returning to the process. You may find this an appropriate setting if any of your applications use the consultation pause function described in the section on Message Switching.

The default is 900 SECONDS/15 MINUTES.

VERCHK

VERCHK=[YES|NO]

If the VERCHK=YES is specified, MAES will compare the version of MAES against that of the transaction driver (that is, AAES or IAESTD) attempting to initiate a consultation. If the versions do not match, MAES rejects the consultation request.

In some circumstances, an earlier or later version of a transaction driver will function properly with the current version of MAES. However, if an application depends on the presence or absence of certain features in the transaction driver, this switch ensures that the two components are compatible.

The default is VERCHK=YES. You can change the value using the MODIFY command.

WTOR

WTOR=[YES|NO]

By specifying WTOR=YES, the following message will be displayed on the operator console:

MAES - ENTER 'T' TO TERMINATE MULTI-TASKING AES

By specifying WTOR=NO, the WTOR will not be displayed. The default is YES.

More Information:

View the MAES Log (see page 213)

Employ Authorization Exit Routines (see page 188)

Dynamically Modify MAES Execution Parameters

Use the z/OS MODIFY command to modify MAES runtime parameters while MAES is running. For example, modify the MAESLOG runtime parameter by typing the following statement on the z/OS console:

```
modify jobname,MAESLOG=YES
```

Where *jobname* is the name of the MAES job. MAESLOG=YES instructs MAES to write log messages to the MAES log.

Specify parameters by typing the keyword followed by an equal sign (=), and then the value for the parameter, as in this example:

```
DEBUGKB=NO,MAESLOG=YES
```

Separate multiple parameters with a comma. If the string of parameters is too long for one line, use multiple MODIFY commands, starting each MODIFY command on a new line.

The following MAES runtime parameters have values that you can change using the z/OS MODIFY command. (See the descriptions of these parameters earlier in this chapter.)

- ADSNAME
- AESNAME
- CPUTIME
- DEBUGKB
- MAESLOG
- MAXABEND
- PMEMBER
- ROUTCDE
- ESTAE
- TASKMEM
- TIME
- STATINT
- VERCHK

You will not receive an error message if you submit an incorrect or misspelled runtime parameter using the MODIFY command. MAES will simply ignore it.

More Information:

View the MAES Log (see page 213)

Specify MAES DD Statements

The DD statements described in this topic apply to all versions of CA Aion BRE.

MAESACCT

```
//MAESACCT DD DSN=&TPHLQ..ACCT,DISP=MOD
```

MAESACCT is used to hold MAES accounting records.

MAESAPRM

```
//MAESAPRM DD DSN=&HLQ.AION.LEPARMS,DISP=SHR
```

The MAESAPRM DD statement specifies a sequential data set that contains any LE run-time parameters that are necessary or desired for the execution of CA Aion BRE applications.

The MAESAPRM data set must contain fixed length, 80-character records without sequence numbers in columns 73 through 80. For example:

```
DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PS)
```

The only time an LE parameter is required for the execution of a CA Aion BRE application is when the application is built with the XPLINK option. In all other cases, it is not required to specify LE options for a CA Aion BRE application.

However, there are a number of times when an application might benefit from the specification of LE parameters. For example, IBM recommends that XPLINK applications specify the HEAPP option in order to obtain the best performance. Also, both XPLINK and non-XPLINK applications can frequently benefit from the specification of the STACK and HEAP parameters to increase the default storage allocations. Finally, there are a number of LE options such as RPTOPTS and RPTSTG that may be useful in a test environment.

When MAES is initialized, it attempts to open MAESAPRM. If the DD statement does not exist, no LE parms are loaded and initialization continues. However, if the DD statement is present, MAES processes the data set and records the LE parameters for later use.

The record format is:

APPNAME='PARMS'

APPNAME

The name of the CA Aion application. *APPNAME* must begin in column 1 and each *APPNAME* must be specified on a separate card.

PARMS

A contiguous stream of LE run-time options separated by commas.

Embedded blanks are not allowed and the application name must be immediately followed by an equal sign and a single quote.

The LE parameter specification for each application can be up to 5 cards long. After the first card containing *APPNAME*, the parameter string can be continued onto four additional cards in one continuous stream of data. When the LE runtime parameters extend beyond the end of a card, all columns up to and including column 80 must contain data. The parameter specification continues in column 1 of the following card and continues until it is terminated by a closing single quote.

For example, the contents of the MAESAPRM dataset might look like the following:

```
APPNAMEA='XPLINK(ON),HEAPP(ON),HEAP(1M)'
APPNAMEB='RPTSTG(ON),RPTOPTS(ON)'
APPNAMEC='HEAPP(ON,32,4,120,3,232,1,352,1,480,1,1440,2)'
APPNAMED='STACK(128K,128K,ANYWHERE,KEEP,512K,128K)'
```

```
//MAESMTAR  DD DSN=&TPHLQ..MAESMTAR.DATA,DISP=SHR
```

MAESLUNM

```
DCB=(RECFM=FB,LRECL=64,BLKSIZE=3072)
//MAESLUNM DD DSN=&MAESHLQ..MAESPARM(&LUNAMES),DISP=SHR
```

MAESLUNM specifies a sequential data set that describes the potential session partners. If the DD statement does not exist, MAES cannot run. You must define the characteristics of *each* possible MAES session partner.

The MAESLUNM data set must contain fixed length, 80-character records without sequence numbers in columns 73 through 80. For example:

```
DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PS)
```

The MAESLUNM data set is a series of source statements in the form:

```
KEYWORD(value) KEYWORD(value) -
KEYWORD(value)
```

The keyword can start in any column as long as the keyword and its corresponding value fit on the same line. Each keyword must have a corresponding value. A definition can be continued on a subsequent line by ending the preceding line with a hyphen (-).

The following list describes the supported keywords and values for the MAESLUNM data set. The keyword is required unless the word Optional appears in parentheses next to the keyword.

LUNAME

The keyword value is the application name (APPLID) of the session partner as specified in the VTAM tables. The following list describes the LU name you can specify for the given partner:

- **CICS**-Is the VTAM node name of the CICS/VS region.
- **IMS**-Is the LU name specified by MAESMON, TLOG or an IMS client program. For IMS client programs accessing apps via ACQUIRE, the LUNAME value can be a generic LU name. Generic LU name entries are five characters, with the fifth character being an asterisk (*). The first four characters contain the value of the MAESCLNT system symbol defined on the Z/OS system upon which the IMS client is being executed. A generic LU name entry must be made in the MAESLUNM file for each concurrent IMS client from each Z/OS system. The generic LU name entries for each client Z/OS system are identical.
- For example, if 'IMS1' is defined on one client Z/OS system as the MAESCLNT system symbol value and support is required for three concurrent IMS clients from that Z/OS system, the following entries would be made in the MAESLUNM file:

```
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
```

These generic LU name entries must be made in the MAESLUNM file of each MAES system participating in the VTAM Generic Resource Pool specified in the MAES field of the IMS client's ACQUIRE call. If the IMS clients are executed on more than one Z/OS system, additional entries must be made in the MAESLUNM file(s). For example, if the IMS clients are executed on two Z/OS systems having 'IMS1' and 'IMS2' defined as their MAESCLNT system symbol values and support is required for three concurrent IMS clients from each Z/OS system, the following entries would be made in the MAESLUNM file of each MAES system in the VTAM Generic Resource Pool:

```
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS1*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS2*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS2*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
LUNAME(IMS2*)    MODENAME(AIONMODE) SYSTYPE(IMS)  ACQUIRE(NO)
```

- **MAES**-Is the node name of the partner MAES region.
- **WINCLIENT**-Is used for Windows SNA clients using independent LU6.2. (Windows clients using dependent LU6.2 must use the IMS LU name.)

ACQUIRE (Optional)

The keyword value is either YES or NO. If you specify ACQUIRE(YES), MAES attempts to bind a session with the session partner at initialization. If the bind fails, MAES continues, and any binding must be done by the session partner or through the MAES monitor knowledge base. If ACQUIRE(NO), MAES leaves the session unbound and expects the session partner to initiate the session later. Default is ACQUIRE(NO).

Notes:

- (CICS Clients) ACQUIRE(NO) must be either specified or defaulted for CICS regions that connect to MAES regions using VTAM Generic Resources (the MAES region's GRNAME value). Specifying ACQUIRE(NO) ensures that CICS initiates the connection, which engages VTAM Generic Resources to determine the MAES system to be used. ACQUIRE(YES) should be specified for CICS regions that connect to a MAES region using its NODENAME value.
- (IMS Clients) Since connections to IMS clients only exist when the IMS client program is executing in the IMS Region, ACQUIRE(NO) must be either specified or defaulted for IMS connections (the IMS client program will initiate the connection).

ALIAS (Optional)

The keyword value is any alternate name for this logical unit for MAES-to-MAES communication. Currently, remote knowledge base support is the only function that can take advantage of this feature. It provides the user a means of isolating a remote knowledge base application from the actual LU name given to the partner MAES.

Note: The ALIAS names for the logical units are scanned first. For example, if a second MAES VTAM name is ALTMAES and the same name is used as the third MAES alias, the second MAES cannot be accessed by the name ALTMAES.

MAX (Optional)

The keyword value specifies the maximum number of sessions that can be acquired to bind this MAES with its session partners. If the session partners specify different maximums, the value used is equal to the lower maximum value. If, for instance, a second MAES is bound, and a maximum of five sessions is specified, and the first MAES specifies 10 sessions, only five sessions are bound. If MAX is not specified, the originating partner's MAX value is used. If the MAX value is reached and MAES receives another BIND request, the bind is rejected.

Note: Specify this keyword only for MAES partners.

WIN (Optional)

The keyword value specifies the number of sessions that this MAES owns for initiating transactions with the partner. The actual number of sessions this MAES owns depends on the corresponding specification in the session partner. In general, if the specifications for MAX and WIN differ between two potential partners, the first one initialized determines the number of sessions owned by each. Where the WIN is less than the MAX, at least one session is allocated to the partner. For remote knowledge base partners connecting one MAES to another MAES, WIN must be greater than zero.

Note: Specify this keyword only for MAES partners.

MODENAME

The keyword value is the VTAM *modename* table entry MAES uses when trying to bind with the session partner at initialization. There is no default.

SYSTYPE

The keyword value generically specifies the session partner's system or program type. The possible values are CICS, IMS, INDEPENDENT, WINCLIENT and MAES. MAES uses this parameter to establish the correct protocol for binding with its session partner. You must specify this parameter.

For example, in the following definition, there are five session partners:

LUNAME(**PCLU0**) MODENAME(AIONMODE) SYSTYPE(WINCLIENT) ACQUIRE(YES) MAX(10) WIN(0)

LUNAME(**TSOLU1**) MODENAME(AIONMODE) SYSTYPE(INDEPENDENT) ACQUIRE(NO)
LUNAME(**IMSLU2**) MODENAME(AIONMODE) SYSTYPE(IMS) ACQUIRE(NO)
LUNAME(**CICSLU3**) MODENAME(AIONMODE) SYSTYPE(CICS) ACQUIRE(NO)
LUNAME(**CICSLU4**) MODENAME(AIONMODE) SYSTYPE(CICS) ACQUIRE(YES)

- The first session partner is an SNA Windows client program using an independent LU6.2. This partner is identified to the VTAM tables as PCLU0.
- The second session partner, called TSOLU1, is an IMS Message Processing Program (MPP) using the TSO transaction driver.
- The third session partner, called IMSLU2, is an IMS Message Processing program using the IMS transaction driver.
- The fourth session partner is a CICS/VS system identified to the VTAM tables as CICSLU3. CICSLU3's connection definition for MAES specifies a VTAM Generic Resource ID (MAES's GRNAME value). In this case, CICS should initialize the session with MAES so that VTAM Generic Resources can select the best MAES system to provide Aion application services.
- The fifth session partner is a CICS/VS system identified to the VTAM tables as CICSLU4. CICSLU4's connection definition for MAES specifies a VTAM APPL ID (MAES's NODENAME value). In this case, MAES can initialize the session with CICSLU4 at startup time because it is the only MAES system which can provide Aion application services for this connection.

In all cases, the mode table entry is AIONMODE.

MAESMTAR

The MAESMTAR data set is used to hold the data for the MAES Timing Analysis Report. The MAESMTAR data set attributes are:

```
DCB=(RECFM=FB,LRECL=153,BLKSIZE=3060)
//MAESLOG  DD SYSOUT=*
```

MAESLOG SYSOUT begins to accumulate runtime status information from the time that MAES successfully starts.

Important! If the MAESLOG DD statement is missing, MAES cannot run.

```
//SESSLOG  DD SYSOUT=*
```

This statement describes a sequential output data set that contains the log of send/receive activity between MAES and its partners.

```
//STEPLIB  DD DSN=&MAESHLQ..LOAD,DISP=SHR
//         DD DSN=&PRODHLQ..LOAD,DISP=SHR
//         DD DSN=&IBMHLQ..SCEERUN,DISP=SHR
```

This statement specifies the name of one or more libraries that contain the Aion/MAES load modules and IBM C/C++ runtime modules. These statements are required if the load modules are not in a library specified in your installation's linklist (LNKLST*nn*).

The IBM C/C++ runtime libraries are required for execution. They are distributed with the IBM C/C++ product.

If you are accessing DB2 from Aion components in MAES, the standard DB2 library (SDSNLOAD) and the DB2 exit library (SDSNEXIT) must also be specified in the STEPLIB.

```
//MAESHPO  DD DSN=&USERHLQ..MAES.LOAD,DISP=SHR
//         DD DSN=&USERHLQ..LOAD,DISP=SHR
```

You should place Aion components, the MAES monitor (MONITOR) and any other modules they requirel in the MAESHPO concatenation..

If you include the MAESHPO DD statement, all program preload requests, external process object requests, and subtask ATTACH requests are executed with this library or library concatenation as the first source.

If a session partner attempts to bind a session with MAES and there is no entry in the MAESLUNM file, the bind is rejected.

MAESPRLD

```
//MAESPRLD DD DSN=&HLQ.ADS..PRELOAD,DISP=SHR
```

The MAESPRLD DD statement specifies a sequential data set that contains the names of load modules that are to be preloaded when MAES initializes.

The MAESPRLD data set must contain fixed length, 80-character records without sequence numbers in columns 73 through 80. For example:

```
DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PS)
```

MAES and its subtasking components dynamically load and delete modules during the course of execution. For example, the OAES load module is brought into memory if a copy does not already exist. That copy is deleted when the consultation is completed.

If the system does not have a consistent level of transaction activity, a new copy of OAES might need to be loaded frequently, causing a slow response time. You can avoid this problem by preloading reentrant modules (which are provided by CA for your installation).

When MAES is initialized, it attempts to open MAESPRLD. If the DD statement does not exist, load modules are not preloaded and initialization continues. However, if the DD statement is present, MAES processes the data set and preloads the specified load modules.

The record format is:

progname,*progname*(*parm*),...,*progname*

progname

the program name.

parm

An optional parameter.

Embedded blanks are not allowed, and a program name must be separated from the next by a comma. The final program name is followed by a blank. If there are more names than can fit in one 80-character record, a comma and blank after the final name in a record indicate that the next record needs to be read and processed.

Note: The first program name must begin in column 1 of the record.

For example, a program preload list might include OAES, MONITOR and TLOG. You would specify the following:

```
OAES,MONITOR,TLOG
```

Preloaded modules must be reentrant. Otherwise, new copies are still loaded when called. Do not preload the MAES load module since it is loaded when the MAES region is started and runs continuously while the MAES region is active.

Other modules that can be preloaded are your own reentrant external programs. Preloading heavily used external routines can improve performance.

Currently, there are 2 parameters you can supply in *parm*, PRIORITY and COPIES. The PRIORITY value must be a value from 1 to 15, inclusive. The lower the value, the lower the dispatching priority of the application. The default value is 8. The COPIES value must be a value from 0 to 999, inclusive. It specifies the number of copies of an Aion application to pre-initialize. The default value is 0. Only CA Aion applications accessed via an ACQUIRE request can be pre-initialized. Attempts to pre-initialize any other programs will result in error conditions.

Suppose you wanted to run an application called AUDIT before running an application called PAYROLL and you wanted to pre-initialize 5 copies of AUDIT. You would specify the following:

```
PAYROLL(PRIORITY=14),AUDIT(PRIORITY=15,COPIES=5)
```

Notes:

- You can only establish execution priority on OAES, TLOG, MONITOR, and CA Aion BRE applications which are accessed via an ACQUIRE request. All CA Aion BRE applications which are not accessed via an ACQUIRE request execute at the same priority, namely the priority associated with OAES. It is not necessary to pre-initialize an app in order to specify an execution priority. To specify a priority for a CA Aion BRE application which will only be dynamically loaded, specify COPIES=0 or do not specify the COPIES parameter. If no execution priorities are specified, all programs will be executed at the same priority.
- TCP/IP component DLLs (and any included library DLLs) must be preloaded if they are included in the MAESHPO DD concatenation.

MAESSTAT

```
DCB=(RECFM=FB,LRECL=208,BLKSIZE=4160)
//MAESSTAT DD DSN=&TPHLQ..STAT,DISP=MOD
```

MAESSTAT is used to hold MAES statistical records.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

Run the MAES Monitor (see page 182)

View the MAES Log

You can view and troubleshoot MAES internal activity and functions using the MAES Log viewer. You can specify which logging information is displayed using the MAES Log Profile.

View MAES Activity

You can configure MAES to maintain a log of all MAES activity, including that of components and subtasks running in MAES. The contents of this log can be helpful when troubleshooting problems with the following MAES events:

- MAES startup
- Client session activity
- VTAM messaging
- Aion component server errors
- Client function call requests
- Server/client MAES messages
- Memory buffer assignment

Aion provides a TSO (ISPF-based) log viewer application to view the MAES log. You can filter or otherwise customize the active log display, and you can view archived logs. To view the active log, the viewer application uses a VTAM connection to MAES.

MAES Log Related Parameters

The following are the parameters that are used by the MAES log:

MAESLOG

Specifies whether to generate a MAES log file and, optionally, limit the range of error message numbers displayed.

DIVLOGDS

Specifies an z/OS-style data set name for the VSAM data set in which to store the MAES log.

DIVLOGSZ

Specifies the size of the MAES log, in 4K blocks (4096 bytes).

Each of these parameters is discussed in the paragraphs that follow.

Write Messages to the MAES Log

Use the MAESLOG runtime parameter to specify that MAES messages be written to a MAES log. The MAESLOG parameter has the following syntax:

```
MAESLOG=[YES|NO|LOG_#S]
```

Specify Whether to Write to a MAES Log

By default, MAESLOG is set to YES, specifying not to write messages to the MAES log. Typically, you set the MAESLOG parameter to YES during periods of testing and debugging, and set MAESLOG to NO in a stable production environment. You can dynamically switch the MAESLOG parameter between YES and NO using the Z/OS MODIFY command.

When set to YES, MAES writes the MAES log to SYSOUT (by way of the MAESLOG DD) during the initialization phase. All messages regarding the startup phase are in the MAES job output. Once the VTAM ACB has been properly opened and MAES is ready for logon, the internal logging starts. Subsequent messages are written to the DIVLOG.

Note: If the initialization fails, MAES terminates with return code 8 and generates an error message in the job output.

Limit the Range of Error Messages

If you want to limit the range of error messages produced, you can specify a range of log numbers for MAES log to show, if they occur. For example, the following command specifies that MAES only write error messages having the codes 1-1000 and 3000-4000 to the MAES log:

```
MAESLOG=(1-1000,3000-4000)
```

You can change this value using the MODIFY command. However, if you wish to subtract messages numbers from the set to be printed, you must first specify:

```
MAESLOG=NO
```

Then, specify all the numbers you wish to be printed. Otherwise, all modifications add to the list.

Specify the Data Set Name for the Log File

Use the DIVLOGDS parameter to specify a z/OS-style data set name for the VSAM linear data set to contain the MAES log. The maximum length of this value is 42 characters, and the last qualifier must be six characters or less (since MAES appends a two-character numeric suffix to the name). If the data set already exists, it is cleared and reused. Otherwise, a new data set is created once MAES starts.

Specify the Size of the MAES Log File

Use the DIVLOGSZ parameter to specify the size of the MAES log in 4K blocks (4096 bytes).

Note: This number must be a multiple of 16. If you specify a number that is not a multiple of 16, MAES internally converts it (for example, 500 becomes 496).

One 4KB block can hold 32 messages (for example, a DIVLOGSZ of 496 holds 15872 messages).

Switch Log Files

At startup, MAES begins logging to the first data set specified by the DIVLOGDS parameter. When that data set becomes full (reaches the size specified by DIVLOGSZ), it is closed and the next data set is created (DIVLOGDS with a 01 suffix). MAES can create up to 100 log data sets.

Each time a log switch occurs, MAES writes the following console message:

```
MAES475 MAESLOG SWITCH, CLOSED LOG DATASET dsname
```

Using the DLOG program, an automation product can use this message to archive the MAES log data set.

Copy Log File Archives

MAES Log file archives can be copied to SYSOUT or sequential files using the following JCL. Use the PARM option to specify the DIVLOG file name.

```
//JOBNAME JOB (99999),'AION',MSGCLASS=X
//*
//DUMPLOG  EXEC PGM=DLOG,PARM='AION.MAES.MLOG00'
//STEPLIB  DD DISP=SHR,DSN=AION.PROD.LOAD
//SYSPRINT DD SYSOUT=*
//*
```

Record Trace File Execution in the MAES Log

Using the BABUILD JCL procedure, you specify the COMPONENT TRACE parameter to write an application trace to the MAES log during compiled execution (in addition to the other MAES messages).

More Information:

Specify Execution Trace Output (see page 76)

Run the MAES Log Viewer

Aion provides a CLIST named TLOG, which you submit to invoke the MAES Log viewer. This is an example:

```
TSO TLOG
```

Note: The TLOG CLIST can be found in <aionhlq>.CLIST dataset.

Note: Prior to running the MAES Log viewer, you must have defined the local and MAES LU names to VTAM, and you must have defined the local LU name in the MAESLUNM table.

To run the MAES Log viewer

1. Submit the TLOG CLIST to start the MAES Log viewer.

   The Aion MAES Log Profile panel appears.

```
 --------------------------- Aion MAES Log Profile ---------------------------

 Command ===


 Local LU Name : AION
 MAES  LU Name : MAES

 Do you wish to filter your display (Y or N):  N

 Please select (s) the columns to display and their size:

 Column        Start   End             Color   Description
 S  Seq#       01      08 (01-08)      BLUE    Message sequence number
    Date               01      07 (01-07)      BLUE    Date the message was
written
 S  Time       01      11 (01-11)      BLUE    Time the message was written
    Userid     01      08 (01-08)      BLUE    User who started the program
    Program    01      08 (01-08)      BLUE    Program that issued the message
    Source     01      08 (01-08)      BLUE    Function that issued the message
    Type                               BLUE    Type of message
    Msgid      01      08 (01-08)      BLUE    Message identifier, if applicable
 s  Text       01      80              GREEN   Message Text (Always displayed)

  F1=HELP      F2=SPLIT     F3=END       F4=RETURN    F5=RFIND     F6=RCHANGE
  F7=UP        F8=DOWN      F9=SWAP     F10=LEFT     F11=RIGHT    F12=RETRIEVE
```

1. Complete the following fields:

   Local LU Name

   Specifies any available VTAM LU6.2 name type, such as AION.

   MAES LU Name

   Specifies the MAES node name specified in DEFPARMS.

2. The MAES Log Profile panel contains a list of all columns that you can select for display.

   Insert the letter **s** next to each column you want displayed when viewing the MAES log. These columns are described in the following table:

   Seq#

   The message sequence number

   Date

   The date the message was written

   Time

   The time the message was written

   Userid

   The user who started the program

   Program

   The program that issued the message

   Source

   The function that issued the message

   Type

   The type of message

   MsgID

   The log message identifier, if applicable. Log messages from 1 to 9999 are informational only. Log messages 10000 and higher indicate error conditions that require corrective action.

   Text

   The message text (required)

   Note: You cannot filter out the Text column; it always displays by default.

3. For each of the columns, you can specify a Start value and End value. Use the Start and End values to display a portion of a column. For example, to see just the first four characters of the user ID, specify **00** in the Start field and **04** in the End field.

4. Choose the color of the display (green, white, turquoise, pink, yellow, blue, or red).

   The default is green.

5. Indicate the screen to show at startup.

6. To display the MAES log for the specified MAES region, press Enter or PF3.

```
SEQ#     DATE    TIME         USERID   SOURCE   T PROGRAM  MSGID    TEXT

00000106 2001054 14:54:47.94           DB2COMP  I OAES     OAES0001 Sending res
00000107 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Function re
00000108 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Function po
00000109 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 interface_r
00000110 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Parm # 1 is
00000111 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Allocated 1
00000112 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Parm # 2 is
00000113 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Parm # 3 is
00000114 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Built param
00000115 2001054 14:54:48.03           DB2COMP  I OAES     OAES0001 Calling int
00000116 2001054 14:54:48.03           DB2COMP  T OAES     RETRC001 0.09    --e
00000117 2001054 14:54:48.03           DB2COMP  T OAES     RETRC001 0.09      :
00000118 2001054 14:54:48.04           DB2COMP  T OAES     RETRC001 0.09      :
00000119 2001054 14:54:48.04           DB2COMP  T OAES     RETRC001 0.09      :
00000120 2001054 14:54:48.09           DB2COMP  T OAES     RETRC001 0.10      :
00000121 2001054 14:54:48.10           DB2COMP  T OAES     RETRC001 0.10      :
00000122 2001054 14:54:48.10           DB2COMP  T OAES     RETRC001 0.10      :
00000123 2001054 14:54:48.10           DB2COMP  T OAES     RETRC001 0.10      :
00000124 2001054 14:54:48.11           DB2COMP  T OAES     RETRC001 0.11      :
00000125 2001054 14:54:48.11           DB2COMP  T OAES     RETRC001 0.11      :
00000126 2001054 14:54:48.11           DB2COMP  T OAES     RETRC001 0.11      -
00000127 2001054 14:54:48.12           DB2COMP  T OAES     RETRC001 0.11
00000128 2001054 14:54:48.12           DB2COMP  T OAES     RETRC001 0.11
00000129 2001054 14:54:48.12           DB2COMP  T OAES     RETRC001 0.11
00000130 2001054 14:54:48.13           DB2COMP  T OAES     RETRC001 0.11
00000131 2001054 14:54:48.13           DB2COMP  T OAES     RETRC001 0.11      -
00000132 2001054 14:54:48.13           DB2COMP  T OAES     RETRC001 0.12      :
00000133 2001054 14:54:48.13           DB2COMP  T OAES     RETRC001 0.12    --c
00000134 2001054 14:54:48.13           DB2COMP  I OAES     OAES0001 interface_r
00000135 2001054 14:54:48.13           DB2COMP  I OAES     OAES0001 Sending res
00000136 2001054 14:54:48.16           DB2COMP  I OAES     OAES0001 Cleanup rec
00000137 2001054 14:54:48.16           DB2COMP  I OAES     OAES0001 Freeing 307
00000138 2001054 14:54:48.16           DB2COMP  I OAES     OAES0001 Terminating
00000139 2001054 14:54:56.44 PDDSH     VLOG     I VLOG     VLOG0001 VLOG: Start
00000140 2001054 14:57:10.62 PDDSH     VLOG     I VLOG     VLOG0002 VLOG: Clean
00000141 2001054 14:58:18.38 PDDSH     VLOG     I VLOG     VLOG0001 VLOG: Start
00000142 2001054 15:01:10.42 PDDSH     VLOG     I VLOG     VLOG0002 VLOG: Clean
```

The MAES log display includes the columns you selected on the MAES Log Profile panel. The number of messages stored in the log is determined by the value of the DIVLOGSZ parameter.

Note: If you specified to generate an Aion Execution Trace, the MAES log includes trace output messages and system messages.

More Information:

Specify MAES DD Statements (see page 204)

Messages and Codes (see page 259)

Perform Tasks from the MAES Log Panel

You can type commands on the command line of the MAES Log Profile panel to perform the following tasks:

- Find an occurrence of a specific value.
- Find the next or previous occurrence of that value.
- Filter the MAES log display.
- Return to the MAES Log Profile panel.

The instructions for performing these tasks are provided in the following paragraphs.

Find a Value in the MAES Log

You can search for a specific value in the contents of an open MAES log.

To find a value in the MAES log

1. On the MAES Log Profile panel command line, enter **find val.**

   where *val* is the string or number you want to find.

2. Press Enter.

   The cursor moves from its current location to the next occurrence of val.

3. To search for the next occurrence, again enter **find val** on the command line and press Enter.

   The cursor moves forward to the next occurrence of val.

4. You can also use FIND PREV commands to find previous occurrences.

Filter the Display

To restrict which values are displayed, type **y** next to the Enable MAES Log Filtering option on the MAES Log Profile panel. When you press Enter, you are prompted to specify filter values for one or more columns.

On the Filtering panel, specify the following information for each column that you want to use in filtering the MAES log display:

- Type **s** in the column's Action field.
- Specify the value for the column you want to retrieve.
- Press F3 to display the filtered log entries.

Aion combines the values in a logical AND combination and conducts the search in the displayed order of the columns. CA Aion BRE begins searching the MAES log for the value specified in the first selected column. Only if it finds a match will the search begin for the next column selected, and so on.

To return to the Filtering panel from the Log Output panel, enter **filter** on the command line.

Return to the Profile

From the MAES Log viewer display, enter **profile** on the command line to return to the Profile panel.

Customize the Display

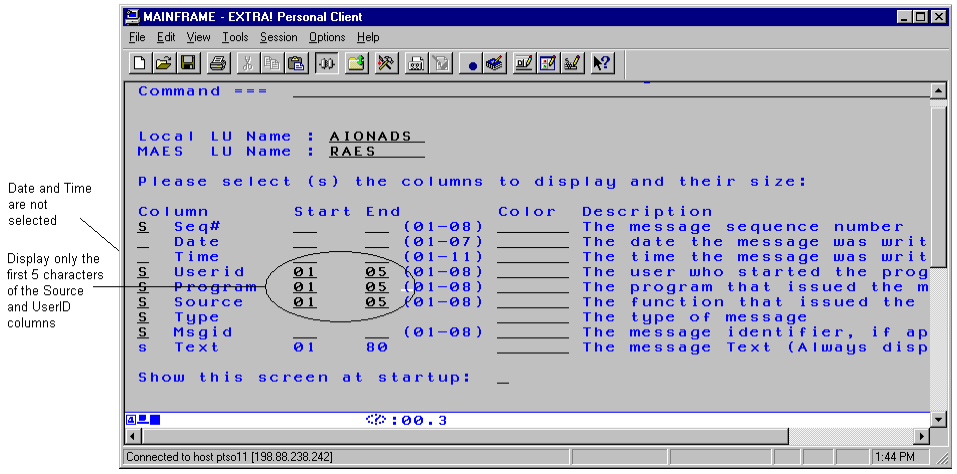
To customize the display you must modify the MAES Log Profile.

To customize the display

1. Specify the columns to display, and how many characters to display in each column.
2. From the MAEL Log viewer, type **profile** on the command line to return to the MAES Log Profile.
3. Do one of the following:
   a. Select or clear columns.
   b. Specify the colors or character widths for columns.

Example:

You might use the customization in this diagram to allocate more display space for the Message Text column. This profile specifies to suppress the Date and Time columns, and to restrict the width of the Userid, Program, and Source columns.



Example: Produces the following MAES Log output

```
SEQ#     USERI SOURC T PROGR MSGID    TEXT

00000408       MAES  M MAESA ADS00001
00000409       MAES  M MAESA ADS00001 127F1EA0  127F2B24 00000000 127F1EE8 000
00000410       MAES  M MAESA ADS00001 127F1EB0  00280000 012C0000 01310000 000
00000411       MAES  M MAESA ADS00001 127F1EC0  D6D5C1C4 E2400000 00000000 000
00000412       MAES  M MAESA ADS00001 127F1ED0  00000000 00000000 00000000 000
00000413       MAES  M MAESA ADS00001 127F1EE0  00000000 00000000
00000414       MAES  M MAESA ADS00001
00000415       MAES  M MAESA ADS00001
00000416       MAES  M MAESA ADS00001 127F2B24  FF0903E3 C4D1E4C4 E8404005 02E
00000417       MAES  M MAESA ADS00001 127F2B34  C7000000 00000000 00000000 000
00000418       MAES  M MAESA ADS00001 127F2B44  00000000 00000000
00000419       MAES  M MAESA ADS00001
00000420       MAES  M MAESB ADS01000 FPL
00000421       MAES  M MAESB ADS01000
00000422       MAES  M MAESB ADS01000 00029EC8  00000003 00029EE4 00029ED8 927
00000423       MAES  M MAESB ADS01000 00029ED8  00050002 0006000C 00040000 000
00000424       MAES  M MAESB ADS01000
00000425       MAES  M MAESN ADS01002 STARTING LOG VIEWER TASK
00000426       MAES  M MAESN ADS01002
00000427       MAES  M MAESN ADS01002 127F1B5C  E5D3D6C7 40404040
00000428       MAES  M MAESN ADS01002
00000429       MAES  M MAESN ADS01003 CONSULTATION STARTED
00000430       MAES  M MAESN ADS01003 MESSAGE INFORMATION FOLLOWS
00000431       MAES  M MAESN ADS01003
00000432       MAES  M MAESN ADS01003 00013144  FFFFFFFF
00000433       MAES  M MAESN ADS01003
00000434       MAES  M MAESR ADS02014 GENERAL
00000435       MAES  M MAESR ADS02014 VTAM RPL FOLLOWS
00000436       MAES  M MAESR ADS02014
00000437       MAES  M MAESR ADS02014 127F2980  00202370 80015534 000156F8 000
```

Note: The USERID, SOURCE, and PROGRAM fields display only the first five characters, and that the DATE and TIME fields do not display at all.

Produce Client Communication Messages

The MAES Log shows client communication messages from OAES. Since these messages do not have message codes assigned to them, they are written to the MAES Log if the MAES Log is activated (MAESLOG=YES or MAESLOG=*msg codes*).

To specify that the MAES Log display only client messages, including Execution Trace, use the following command:

```
MAESLOG=(2-3)
```

The following is an example of filtered MAES Log output, showing only client messages (including application trace statements):

```
Parm # 8 is DATE: 2039/12/31 at B36C2

Parm # 9 is INT:  12B31EA8
Parm # 10 is REAL:  0.000000
Parm # 11 is LPSTR, Length(1024)
Parm # 12 is BOOL:  12B31ED8
Parm # 13 is INT:  12B31EE8
Parm # 14 is REAL:  0.000000
Parm # 15 is LPSTR, Length(1024)
Parm # 16 is BOOL:  12B31F18
Parm # 17 is INT:  12B31F28
Parm # 18 is REAL:  0.000000
Parm # 19 is LPSTR, Length(1024)
Parm # 20 is BOOL:  12B31F58
Parm # 21 is INT:  12B31F68
Parm # 22 is REAL:  0.000000
Parm # 23 is LPSTR, Length(1024)
Built parameter list:
Calling otype_mall_many
0.08    --executing mall_many (otype_00001)
0.08    --completed mall_many
otype_mall_many returned: 11C708
Sending response for otype_mall_many
Function request: otype_Delete
Function pointer: 12A97FF8
Returntype is INT
otype_Delete has 1 parms:
Parm # 1 is INT:  12B31668
Built parameter list:
Calling otype_Delete
0.10    --application terminated
0.10    Aion execution completed:: 2001/02/12 10:11:14
otype_Delete returned: 0
Sending response for otype_Delete
Cleanup received,shutting down.
Freeing 3072 bytes of memory at: 12A83028
Terminating subtask A001
VLOG: Cleanup received, shutting down
```

Terminate MAES

You can use any of the following methods to terminate MAES:

- **During initialization**-MAES generates this WTOR message:

  MAES - ENTER 'T' TO TERMINATE MULTI-TASKING AES

  Enter **t** to terminate MAES.

  Note: The WTOR message is not generated if you have specified NO for the WTOR MAES runtime parameter.

- **From the z/OS console**-Enter the following command to terminate MAES:

  `stop jobname`

  where:

  jobname is the name of the MAES application.

  From the z/OS console, you can cancel the MAES job. MAES intercepts the cancel request and checks that executing subtasks have terminated properly.

  Note: If any consultations are still executing, MAES does not stop. However, if you cancel MAES a second time, MAES stops immediately without waiting for any subtasks to end.

- **From the MAES monitor application**-Select the Kill MAES option from the MAES Monitor Main Menu to terminate the MAES session. When the MAES monitor asks for a confirmation, specify YES to terminate the session.

  You can use the KILLMAES parameter to specify whether a user can terminate MAES from the MAES monitor application.

More Information:

Specify MAES Runtime Parameters and DD Statements (see page 192)

MAES Reports: Gathering Statistics

MAES can capture running subtask statistics at fixed time intervals. The time interval is specified with the STATINT MAES runtime parameter.

Statistics gathering begins when the following requirements are met:

- The value specified by STATINT is greater than zero.
- A DD statement for MAESSTAT has been included in the MAES JCL.

The data set described by MAESSTAT is typically on disk. MAES automatically applies the data set characteristics (DCB attributes). For example, you might specify the MAESSTAT DD statement as follows:

```
//MAESSTAT DD DSN=SYS2.AION.MAES.STATS,DISP=(NEW,CATLG),
// UNIT=SYSDA,SPACE=(CYL,(5,1)), DCB(RECFM=FB,LRECL=64,BLKSIZE=3072)
```

The information gathered includes the amount of virtual memory used by each consultation, and how much of that usage is divided between the program and data processing tasks. This information can be useful for tuning your applications as well as determining the appropriate region size for the MAES address space.

When MAES is started, it initializes a separate task within the address space that sets a timer based on the value of STATINT. When the timer expires, the statistics task writes a record to the MAESSTAT data set reflecting information about the MAES task, and one additional record for each active consultation. The resulting data set can be processed by the AIONSTAT program or by your own program if you wish to produce a customized report or histogram.

Note: A copy of the JCL that executes the AIONSTAT program is provided in the <aionhlq>.MAES.JCL as member STATREP.

A Statistics Report is shown in the following figure:

```
                                AION Multitasking Statistics Report   Page: 1

Mon Feb 15 14:27:24 2001
99/043 14:29:03:72 AC :0000,SU :0000,ID : 0013, Frames : 2307, VTAM not posted
99/043 14:29:03:72 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2416K       36K     1148K      124K      376K
   >16M                         103904K     8228K    19796K     2816K     4960K

99/043 14:29:33:77 AC :0000,SU :0000,ID : 0013, Frames : 2308, VTAM not posted
99/043 14:29:33:77 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2416K       36K     1148K      124K      376K
   >16M                         104040K     8228K    19848K     2816K     4960K

99/043 14:30:06:61 AC :0000,SU :0000,ID : 0013, Frames : 2301, VTAM not posted
99/043 14:30:06:61 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2424K       36K     1152K      124K      376K
   >16M                         104048K     8228K    20496K     2816K     4960K

99/043 14:30:36:67 AC :0000,SU :0000,ID : 0013, Frames : 2301, VTAM not posted
99/043 14:30:36:67 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2424K       36K     1148K      124K      376K
   >16M                         103948K     8228K    20284K     2816K     4960K

99/043 14:31:06:71 AC :0000,SU :0000, ID : 0013, Frames : 2301, VTAM not posted
99/043 14:31:06:71 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2416K       36K     1148K      124K      376K
   >16M                         103360K     8228K    20712K     2816K     4960K

99/043 14:31:36:78 AC :0000,SU :0000,ID : 0013, Frames : 2301, VTAM not posted
99/043 14:31:36:78 **MAES**        CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2416K       36K     1148K      124K      376K
   >16M                         103392K     8228K    21132K     2816K     4960K
99/043 14:38:07:37 MONITOR  A001   CSA      LSQA       SQA      DATA    SYSTEM
   <16M                           2416K       36K     1144K       96K       12K
   >16M                         103340K     8352K    22336K      376K       16K
```

Debug Applications with MAES Internal Trace

MAES maintains an internal trace table that can be used when debugging applications. The internal trace is maintained as a series of wraparound lists. The number of lists is equal to the value of the MAES runtime parameter, MAXTASKS. The number of entries in each list is determined by the value specified for the MAES runtime parameter, ITRACE.

Assume, for example, that you specify the following MAES runtime parameters:

```
MAXTASKS=15,ITRACE=20
MAES will maintain 15 lists of 20 entries each.
```

If ITRACE is set to zero, the internal trace is disabled. The default value for ITRACE is 10. Each entry in a list represents some internal request, such as sending of a message or changing the state of a session.

The internal trace can be captured using the MAES monitor application.

Note: The MAES Monitor CLIST can be found in <aionhlq>.CLIST as member (MAESMON).

To capture an internal trace

1. From the Main Menu, choose Print VTAM activity trace.
2. Specify a SESSLOG DD statement in the MAES JCL to receive the trace output.

   The following example shows sample trace output written to the SESSLOG data set:

```
PRINT REQUEST FROM A972    /AIONDEVO

 SESSIONS WITH AIONQAC
 TRACE TABLE FOR SESS ID 00000000
 MAES ACTIVITY
 NONE
 SAES ACTIVITY
 NONE
 TRACE TABLE FOR SESS ID 00000000
 MAES ACTIVITY
 NONE
 SAES ACTIVITY
 NONE
 SESSIONS WITH AIONDEVO
 TRACE TABLE FOR SESS ID 0300057D
 MAES ACTIVITY
 09.35.44.5 MAESSECS  OPENSEC
   0935445F D4C1C5E2 E2C5C3E2 00000000   * ...^MAESSECS.... *
   2A000000 00000000 00011E3C 00011DFC   * ...............  *
   00000000 0300057D 00001000 80000010   * .......'........ *
   30945080 00801200 00000000 00000000   * .m.............. *
   00000000 00000000 00000000 00000000   * ................ *
   00000000 00000000 00000000 00000000   * ................ *
 09.35.45.0 MAESRECP  RECEIVE (AFTER) FDBK=0000 CEB OIC FMH EX (1) SEQ: 0001
   0935450F D4C1C5E2 D9C5C3D7 00000000   * ....MAESRECP.... *
   23800000 00000000 033FA418 033FB8F0   * ..........u....0 *
   00000000 0300057D 01001004 00200092   * .......'.......k *
   30945080 00801100 06000000 00010000   * .m.............. *
   00000000 00000000 00000000 00000000   * ................ *
   00000000 00000000 00000000 00000000   * ................ *
***  PRINT END ***
```

   This example shows session information, time of day, VTAM return codes, and internal control blocks (to assist technical support).

Extract MAES Resource Accounting Data

Production shops commonly charge the user for the costs of running an application. Typically, these charges are determined from CPU time, I/O requests, and elapsed time consumed by the application. When running Aion, the normal System Management Facility (SMF) data is automatically produced at the time the user logs off from TSO.

In a single-user environment, all CPU, I/O, and elapsed time can be readily isolated. In a multiregion environment, new methods must be developed for tracking the resources used by specific transactions or subtasks.

Currently, there are well-established techniques for charging the user for the costs of running under CICS/VS or IMS/VS. When running Aion in these environments, however, a significant portion of the application executes under control of MAES. The resources used by the transaction driver component are still captured with the existing accounting techniques.

The following paragraphs describe how you can extract data about the resources used by subtasks within the MAES address space.

More Information:

Accounting Record Format (see page 255)

Create MAES Accounting Records

MAES captures information about the resources used by each subtask under its control. This information is captured internally and can be examined dynamically using the MAES monitor knowledge base. Each time a subtask ends, this same information can be sent to a data set by specifying the MAESACCT DD statement in the MAES JCL.

If the ACCT dataset does not exist, specify:

```
//MAESACCT DD DSN=<aionhlq>.SYS2.AIONnn.ACCT,DISP=(NEW,CATLG,DELETE),
// DCB=(RECFM=FB,LRECL=208),UNIT=SYSDA,
//  SPACE=(CYL,(4,1))
```

If the ACCT dataset exists, specify:

```
//MAESACCT DD DSN=<aionhlq>.SYS2.AIONnn.ACCT,DISP=OLD
```

In these examples, the contents of <aionhlq>.SYS2.AION*nn*.ACCT are overwritten when the accounting information is produced. To add new accounting information to the end of this data set, specify the MAESACCT DD statement as follows:

```
//MAESACCT DD DSN=<aionhlq>.SYS2.AIONnn.ACCT,DISP=MOD
```

During initialization, MAES checks to see if the MAESACCT DD statement exists. If MAESACCT is allocated, MAES generates an accounting record every time a subtask terminates.

Aion supplies an AIONACCT program that reads the MAESACCT data set and creates a report. JCL to prepare the accounting report is provided in <aionhlq>.MAES.JCL(ACCTREP). This JCL includes customization descriptions, and instructions for executing the job.

The AIONACCT program formats the record and prints each field in the record with a cumulative total as the last element of the report. A sample of an Accounting Report is shown in the following figure.

```
1  AION Multitasking Accounting Report                      Page :   1
 Mon Feb 15 14:43:44 2009
-------------------------------------------------------------------------------------------------
0User :PDDSH  ,kb :ARGSERV , id :1095 ,LUname :CICSCXC ,code :000(Normal), actions: 75,cpu :488.87
   Start : Feb 12, 2001 15:21:08:20, max task memory : 1024000, 24-bit memory : 0
   End   : Feb 12, 2001 15:21:12:22,   max allocated : 1024000

            VTAM-RUs     Message   Data-reqs  Other-msgs Total bytes
     IN :       227          76          0         151       18576
    OUT :       226           1          0         226        5943
0              States loaded : 0, KB EXCPs : 0, Ext. pgms loaded : 0
0                     Gets        Adds     Updates     Deletes
               VSAM       0           0           0           0
               QSAM       0           0           0           0
                DB2       0           0           0           0
               DL/I       0           0           0           0
               IDMS       0           0           0           0

0User :PDSHI  ,kb :MONITOR , id :1118  ,LUname :CICSCXC ,code :46077(User),actions: 19,cpu :363.83
   Start : Feb 12, 2001 14:38:06:24, max task memory : 1024000, 24-bit memory : 4682
   End   : Feb 12, 2001 15:32:22:64,   max allocated : 881424
               VTAM-RUs     Message   Data-reqs  Other-msgs Total bytes
      IN :        58          19          0          39         907
     OUT :        57          19          0          38       18117
0                States loaded : 0, KB EXCPs : 0, Ext. pgms loaded : 6
0                      Gets        Adds     Updates     Deletes
               VSAM       0           0           0           0
               QSAM       0           0           0           0
                DB2       0           0           0           0
               DL/I       0           0           0           0
               IDMS       0           0           0           0

0User :PDSHI  ,kb :MONITOR ,id :1118  ,LUname :CICSCXC ,code :46077(User),actions : 22,cpu :491.90
   Start : Feb 12, 2001 15:32:50:14, max task memory : 1024000, 24-bit memory : 4682
   End   : Feb 12, 2001 15:44:45:30,   max allocated : 885912
               VTAM-RUs     Message   Data-reqs  Other-msgs Total bytes
       IN :       67          22          0          45        1081
      OUT :       66          22          0          44       20668
0                States loaded : 0, KB EXCPs : 0, Ext. pgms loaded : 7
0                      Gets        Adds     Updates     Deletes
               VSAM         0           0           0           0
               QSAM         0           0           0           0
                DB2         0           0           0           0
               DL/I         0           0           0           0
               IDMS         0           0           0           0

0User :PDSHI  ,kb :MONITOR , id :1118  , LUname :CICSCXC ,code :46077 (User),actions :23, cpu :386.23
  Start : Feb 12, 2001  15:45:03:89, max task memory : 1024000, 24-bit memory : 4682
  End   : Feb 12, 2001 15:50:10:15,   max allocated : 883472
               VTAM-RUs      Message   Data-reqs  Other-msgs Total bytes
      IN :       70          23          0          47        1073
     OUT :       69          23          0          46       22453
0                     States loaded : 0, KB EXCPs : 0, Ext. pgms loaded : 6
0                         Gets        Adds     Updates     Deletes
               VSAM         0           0           0           0
               QSAM         0           0           0           0
                DB2         0           0           0           0
               DL/I         0           0           0           0
               IDMS         0           0           0           0
```

More Information:

The MAES Monitor (see page 183)

Customize the Accounting Routine

A user exit is available within the MAES load module to allow you to customize the accounting routine. Examples of accounting functions you can control by customizing this routine include:

- Direct the accounting record to SMF or another output destination.
- Suppress the accounting record by querying information in the record or elsewhere in the system.
- Modify the contents of the record.

The user module must have MAESUACC as its entry point. The following information is passed to MAESUACC in a standard operating system parameter list:

- Address of the data control block (DCB) that is used for the accounting file
- Address of the accounting record
- Size of the accounting record (the size of the accounting record is 208 bytes and *cannot* be changed)

Note: Register 1 point to the above information upon entry.

For example, consider a user module that writes SMF records. To write SMF records, the user module would receive the accounting record information, reformat it into SMF record format, and write the record to SMF.

On return from the exit, MAES looks at the return code in register 15. If the return code is zero, the accounting record is written normally. If any other value is returned in register 15, the accounting record is not written.

The custom accounting routine can be defined by the following job.

```
//<JOBCARD>
//*
/*JOBPARM SYSAFF=*
//*
//* REPLACE THE FOLLOWING:
//*  <AIONHLQ> AION LIBRARY QUALIFIER
//* <MAESHLQ> MAES LIBRARY QUALIFIER
//*
//ASMA PROC PROG=
//ASM EXEC PGM=ASMA90,REGION=2M,
//  PARM=('NORENT,LIST,DECK,OBJECT')
//*
//SYSPUNCH DD DUMMY
//*
//SYSUT1   DD UNIT=VIO,SPACE=(TRK,(5,5))
//*
//SYSLIB   DD DISP=SHR,DSN=<AIONHLQ>.MACLIB
//         DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=SYS1.AMODGEN,DISP=SHR
//SYSIN    DD DUMMY
//*
//SYSLIN   DD DSNAME=&&OBJ,UNIT=SYSDA,DISP=(NEW,PASS),
// SPACE=(TRK,(1,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//*
//SYSPRINT DD SYSOUT=*
//*
// PEND
//*
//S1 EXEC ASMA,PROG=MAESUACC
//ASM.SYSIN DD *
```

```
*--------------------------------------------------------------------*
*                                                                    *
* MAESUACC                                                           *
*                                                                    *
* INPUT - ADDRESS OF THE ACCOUNTING DCB                              *
*         ADDRESS OF THE FORMATTED ACCOUNTING RECORD                 *
*         LENGTH OF THE ACCOUNTING RECORD                            *
*                                                                    *
* OUTPUT - R15 = 0 IF MAES SHOULD WRITE OUT THE RECORD               *
*          R15 IS NOT 0 IF MAES SHOULD BYPASS WRITING THE RECORD     *
*                                                                    *
*--------------------------------------------------------------------*
MAESUACC SUBSTART REGS=YES
*
* >>> PUT CUSTOM ACCOUNTING PROCESSING HERE... <<<
*
         SUBEND RETCODE           RETURN TO THE CALLER
RETCODE  DC    F'0'
         END
/*
//*
//REMAES EXEC PGM=IEWL,PARM='AMODE=31,RMODE=24,NORENT,NCAL',
//  COND=(4,LT)
//*
//SYSPRINT DD SYSOUT=*
//UACCOBJ   DD DSN=&&OBJ,DISP=(OLD,DELETE)
//MAESLIB DD DSN=<MAESHLQ>.LOAD,DISP=SHR
//SYSLMOD DD DSN=<MAESHLQ>.LOAD,DISP=SHR
//SYSLIN DD *
 INCLUDE UACCOBJ
 INCLUDE MAESLIB(MAES)
 ENTRY MAES
 SETCODE AC(1)
 NAME MAES(R)
/*
```

Perform MAES Timing Analysis

The MAES Timing Analysis function records the amount of time used by executing subtasks during a specified interval. An interval is defined as the time between starting and stopping MAES Timing Analysis.

This information is recorded in the MAES Timing Analysis Report. The primary category of timing information shown on the report is indirect (remote) data access calls. Except for the Total Task CPU Time field, all times in the report are elapsed times.

The MAES Timing Analysis Report contains the information described by the following paragraphs.

Interval Summary

This section shows the following information about the interval:

- Interval duration
- Interval start and stop times
- The number of records processed
- The number of consultations monitored

Complete Consultations

This section shows information about all subtasks that have completed running during the interval (whether or not they started during the interval).

Active Consultations

This section shows information about all subtasks that are still running at the end of the interval (whether or not they started during the interval).

The following is a sample MAES Timing Analysis Report:

```
--------------------------------------------------------------------------------
                      MAES Timing Analysis Report              Page :   1
                      Interval Summary Long Listing
                      Tue Aug  3 08:37:19 2009
 Interval Duration      : 00:02:17.68
 Started                : Aug 3, 1998 08:31:50.78
 Ended                  : Aug 3, 1998 08:34:08.46
 Records Processed      : 67
 Consultations Monitored: 1
 X-Memory               : 0
 VTAM                   : 1
 -------------------------------------------------------------------------------
                      MAES Timing Analysis Report              Page :   2
                      Complete Consultations Long Listing
                      Tue Aug  3 08:37:19 1998
 -------------------------------------------------------------------------------
 Consultation ID   : A001
 UserID            : C3871DM
 Consultation Start: Aug 3, 1998 08:33:30.45
 Consultation End  : Aug 3, 1998 08:34:08.46
 Interval          : 00:00:38.01
 Load Module       : SAES           KB Name: DBMSTST
 MAES Name         : RAES
 Partner LU        : AIONDEVO  Partner Type: CICS  Traffic Type:     VTAM
                                                                     %

Consultation
 Total Transaction Driver Response Time:                       0:00:12.46  32.78%
       Longest : 00:00:10.38
       Shortest: 00:00:00.01
   Status Message Time:                      00:00:00.56
   Remote Report Time:                       00:00:00.00
   Screen I/O Time:                          00:00:00.86
       Longest : 00:00:00.42  Scrn #:    3
       Shortest: 00:00:00.01  Scrn #:    4
   Async API Time:                           00:00:00.58
       Function  :        GetSystemInfo  Time: 00:00:00.06
       Function..:   GetObjectContraint  Time: 00:00:00.02
       Function..:         GetGroupText  Time: 00:00:00.03
       Function  :        GetObjectText  Time: 00:00:00.06
       Function  :       InitObjectList  Time: 00:00:00.03
       Function  :        GetObjectInfo  Time: 00:00:00.04
       Function  :       SetParamValues  Time: 00:00:00.02
       Function  :       SetParamValues  Time: 00:00:00.02
       Function  :       InitObjectList  Time: 00:00:00.05
       Function  :        GetParamValue  Time: 00:00:00.02
       Function  :          SendMessage  Time: 00:00:00.02
       Function  :        GetObjectInfo  Time: 00:00:00.01
       Function  :       SetParamValues  Time: 00:00:00.01
       Function  :       SetParamValues  Time: 00:00:00.01
       Function  :       InitObjectList  Time: 00:00:00.08
       Function  :        GetParamValue  Time: 00:00:00.02
       Function  :          SendMessage  Time: 00:00:00.02
 Remote Data Access Time:                      00:00:11.04
     Total DB2 : 00:00:11.04
       Function  : Select
       Cursor    : CURSE0
       SQL Stmt  : select * from c3871dm.db2test where last_name like 'SMITH%'
       Time      : 00:00:10.38
       Function  : Open
       Cursor    : CURSE0
       SQL Stmt  : select * from c3871dm.db2test where last_name like 'SMITH%'
```

```
       Time      : 00:00:00.03
       Function  : Fetch
       Cursor    : CURSE0
       SQL Stmt  : select * from c3871dm.db2test where last_name like 'SMITH%'
       Time      : 00:00:00.47
       Function  : Fetch
       Cursor    : CURSE0
       SQL Stmt  : select * from c3871dm.db2test where last_name like 'SMITH%'
       Time      : 00:00:00.14
       Function  : Close
       Cursor    : CURSE0
       SQL Stmt  : select * from c3871dm.db2test where last_name like 'SMITH%'
       Time      : 00:00:00.02
    Total DL/I: none
    Total VSAM: none
    Total IDMS: none
     User Pgms: none
Total Wait Time:                                      00:00:17.92  47.15%
      Longest : 00:00:03.00
      Shortest: 00:00:01.15
Total Miscellaneous Message Time:                     00:00:00.00   0.00%
Total Knowledge Base Processing Time:                 00:00:07.63  20.07%
Total Task CPU Time:                                  00:00:00.0130
```

Run Timing Analysis

To run MAES Timing Analysis, specify a MAESMTAR DD statement in the MAES JCL. The data set associated with the MAESMTAR DD statement will contain the information that is gathered during the timing interval.

MAESMTAR data set attributes are:

```
DCB=(RECFM=FB,LRECL=153,BLKSIZE=3060)
```

For example, if you have installed Aion/CICS, you might specify the MAESMTAR DD statement as follows:

```
//MAESMTAR DD DSN=SYS2.MAESMTAR.DATA,DISP=SHR
```

If the MAESMTAR DD statement is included in the MAES JCL, you can use the following methods to activate MAES Timing Analysis:

- Specify MTAR=YES as a MAES runtime parameter when you activate MAES.
- Run the MAES monitor application, choose the Manage Timing Analysis option, and activate MAES Timing Analysis.

More Information:

Run the MAES Monitor (see page 182)

Create a Timing Analysis Report

Use the MAESMTAR program to read the MAESMTAR.DATA data set and create a MAES Timing Analysis Report. For sample MAESMTAR JCL, For samples see MAESMTAR JCL, refer to <aionhlq>.MAES.JCL.

To create a Timing Analysis Report

1. Modify and submit the MAESMTAR JCL. The report output can be sent to a printer or to a data set.

   Note: If you send the report to a data set, use a standard TSO printing protocol, such as IEBGENER, to print the report at a later time.

2. To control the number of lines printed before a page break is inserted and the amount of detail provided in the report, use the following parameters with the MAESMTAR JCL:

   `[LINES=nnnn] [FORM={SHORT|LONG}]`

   where *nnnn* specifies the number of lines to be printed before a page break is inserted.

   Choose either SHORT or LONG to specify the amount of detail to be included in the report. The default LINES value is 61, and the default FORM value is SHORT. When FORM=SHORT, the report shows only the total time for each subtask.

3. If you need more detail in the Remote Data Access Time and the Async API Time fields, specify **FORM=LONG** for a long listing. The report shows the total times for each subtask and the data functions for each access method under the heading, Remote Data Access Time (for each subtask). Using this information, you can determine which remote data acquisitions you need to analyze.

More Information:

Interpret the Long Listing (see page 242)

Interpret the Consultation Listing

The Complete Consultations and Active Consultations listings are shown after the Interval Summary listing. While the information in the Interval Summary listing is somewhat self-explanatory, the Complete Consultations and Active Consultations sections are less intuitive. This section describes the Complete Consultations and Active Consultations sections.

The longest and shortest times are listed for the following:

- Total transaction driver response time
- Screen I/O time
- Asynchronous API time
- Remote data access time (for each access method)
- Total wait time

Note: The longest and shortest remote data access times for each access method are shown only in the short listing.

The fields described in the following paragraphs are included in the Complete Consultations and Active Consultations listings. These fields correspond to the fields in the sample MAES Timing Analysis Report.

Header Information

The header information fields include the following:

Consultation ID

The internal identifier for the consultation.

User ID

The user ID of the person running the consultation.

Consultation Start (or Interval Start)

Specifies the date and time when the consultation started, or when the first record from this consultation was received. If the consultation was already active when MAES timing analysis was started, the consultation start time corresponds with the interval start time.

Consultation End (or Interval End)

For the Completed Consultations listing, contains the date and time when the consultation ended. For the Active Consultations listing, this field contains the date and time when the interval ended.

Interval

Specifies the amount of time that elapsed between the Consultation Start or Interval Start time and the Consultation End or Interval End time.

Load Module

Specifies the name of the executing load module. This can be a SADS or an optimized application name.

MAES Name

Specifies the MAES APPLID.

Partner LU

Specifies the APPLID or cross-memory services partner used for MAES-to-transaction-driver communication.

Partner Type

Specifies the type of partner logical unit.

Traffic Type

Specifies the type of MAES-to-transaction-driver traffic.

Other Consultation Information

The following fields also appear in the Consultation Information listing:

Total Wait Time

Specifies the total time MAES is idle. This field also includes the percentage of the consultation interval during which MAES is idle. MAES is considered *idle* when it is waiting for a user response.

Total Miscellaneous Message Time

Specifies the total time used by unexpected message types. This field also includes the percentage of the subtask interval used by unexpected message types.

Total Knowledge Base Processing Time

Specifies the total elapsed time used by the subtask. This field also includes the percentage of the consultation interval used by the subtask.

Total Task CPU Time

Contains the total amount of CPU time used by the consultation. The CPU time used by the consultation is measured from the beginning of the measured interval to the end of the measured interval.

Interpret the Long Listing

The long listing contains all data from the short listing, as well as additional data to help pinpoint which remote data access and asynchronous AionDS API function call is taking the longest to complete.

In the Remote Data Access Time field, immediately following the total time listed for each remote data access method, is a description of the functions that the knowledge base requested the transaction driver to complete. For each function, the DD name, function name, and the amount of time the function took to complete is shown. The function's operation and data might also be shown.

In the Async API Time field, each function call is listed indicating the function name and the amount of time it took for the function to complete.

MAES Authorized Execution Considerations

MAES, its VTAM server components, and its TCP/IP server components, will optionally operate as APF-authorized z/OS programs. Special capabilities, such as RACF sign-on verification, are only available when these programs are authorized. When MAES is unauthorized, these capabilities will be unavailable.

When MAES is executing as an APF-authorized program, your applications (KBs) and associated programs that are accessed via STEPLIB and MAESHPO DDs will need to be placed in authorized libraries. Your applications (KBs) and programs will be executed in unauthorized contexts. So there is no capability for your applications (KBs) and associated programs to perform authorized activities.

When MAES VTAM capabilities are established in an authorized context, more efficient authorized path VTAM facilities are used. If MAES is unauthorized then this performance improvement is not obtained.

When MAES is APF-Authorized

- More efficient authorized path VTAM communications can be used.
- RACF sign-on verifications can be performed for arriving VTAM and TCP/IP requests.
- VTAM sign-on verifications will not be performed if the MAESPARM is SEC=NONE
- MAES capabilities execute authorized, but customer applications (KBs) and programs execute in unauthorized contexts.

When MAES is Not APF-Authorized

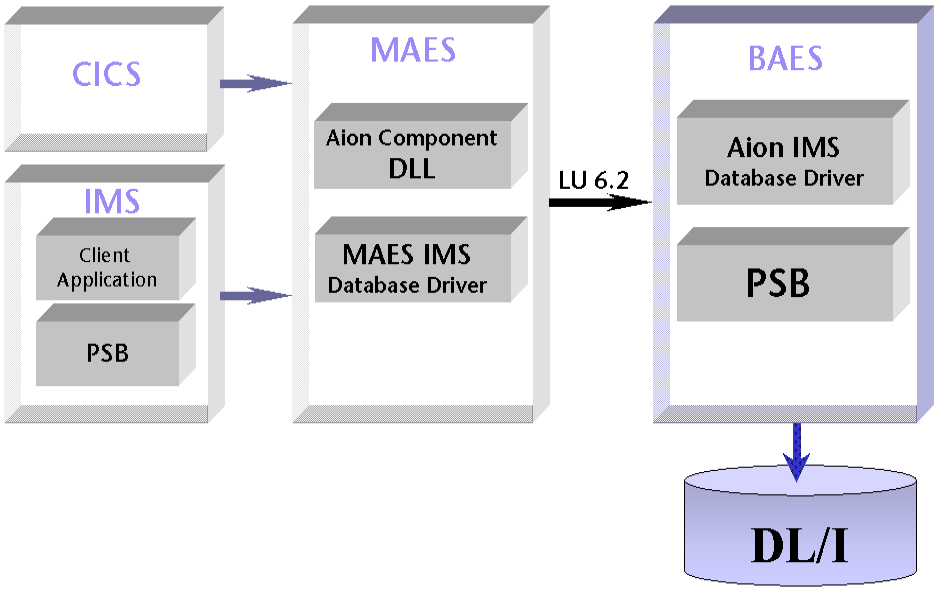
- All TCP/IP sign-on verifications will be rejected.
- VTAM sign-ons will be permitted, only if the MAESPARM is SEC=NONE.
- Authorized path VTAM communications will not be used.

Chapter 6: The BMP/Batch Aion Execution System

BAES is the BMP/Batch Aion Execution System that can be used by Aion applications to access IMS DL/I databases.

This section contains the following topics:

This illustration shows an overview of BAES, MAES, and CA Aion BRE:



Prior to running BAES, the MAES address space has to establish a VTAM LU6.2 session with the BAES region. Once both regions are connected, an Aion component (DLL) running in MAES can access DL/I databases through BAES.

Multiple copies of BAES can run in different BMP regions. Each region is assigned a different VTAM LU 6.2 connection. BAES manages its own VTAM sessions with MAES to satisfy DL/I requests coming from the Aion component. BAES is a multitasking environment. It attaches a data requester task for each incoming request.

Start BAES

BAES capabilities require installation of AION/IMS components. Please contact your installation support staff if these components are absent.

To install BAES, follow the instructions furnished in the *CA Aion BRE Mainframe Installation* Guide. The BAES load module is copied into the <AIONHLQ>.IMS.LOAD PDS.

BAES runs as a job within the operating system. This job can be a started task, or it can be submitted as a batch job.

Execute the BAES Startup JCL

The BAES PROC is used to start BAES. Start BAES by submitting the installed BAES PROC JCL to z/OS. Aion provides the sample JCL that you can use to start BAES. This JCL is supplied in the Aion JCL data set<aionhlq>.MAES.JCL(BAESJCL).

Specify the following variables in the startup JCL to identify your libraries:

Parameter	Description
<*imshlq*>	High-level qualifier for the IMS system library
<*cobolhlq*>	High-level qualifier for the COBOL installation library
<*aionhlq*>	High-level qualifier for the AION module libraries
<*maeshlq*>	High-level qualifier for the AION MAES module library

```
//AIONBAES JOB (ACCT,BIN),'EXECUTE BAES'
//*-------------------------------------------------------------------*
//* *
//* SAMPLE JCL TO EXECUTE THE AION/IMS DL/I DATA ACCESS SUPPORT BMP *
//* *
//* THIS IS USED IN CONJUNCTION WITH THE AION/IMS AND AION FROM *
//* WINDOWS TO ACCESS DL/I DATA FROM AN AION APPLICATION. *
//* *
//IMSRUN EXEC IMSBATCH,MBR=BAES,PSB=IAESPSB,SOUT='*',RGN=5M,
// OPT=C
//G.STEPLIB DD
// DD
// DD DSN=<imshlq>.RESLIB,DISP=SHR
// DD DSN=<cobolhlq>.COB2LIB,DISP=SHR
// DD DSN=<AIONHLQ>.IMS.LOAD,DISP=SHR
// DD DSN=<MAESHLQ>.LOAD,DISP=SHR
//G.SYSPRINT DD SYSOUT=*
//G.MAESLOG DD SYSOUT=*
//G.MAESPARM DD DSN=<MAESHLQ>.MAESPARM,DISP=SHR
//G.MAESLUNM DD DSN=<MAESHLQ>.LUNAMES,DISP=SHR
```

Modify the EXEC Statement

In the EXEC statement, specify the size of the BAES region and any BAES run time parameters.

The following shows an EXEC statement that executes the Aion BAES program in the IMS BMP region using PSB=baespsb to control accesses to DL/I database:

```
//BAESBMP EXEC IMSBATCH,MBR=BAES,PSB=baespsb,SOUT='*',RGN=5M,
// OPT=C
```

MBR=BAES

Is the member name of the BAES load module.

PSB=baespsb

Is the name for BAES.

SOUT='*'

Is the SYSOUT message class.

RGN=5M

Is the region size of IMS BMP.

OPT=C

Indicates that BAES automatically terminates if the IMS control region is not active when this job is started.

Modify the DD Statements

The DD statements described by this topic apply to all versions of CA Aion BRE.

More Information:

Specify BAES Run Time Parameters (see page 250)

//MAESLOG DD SYSOUT=*

The MAESLOG data set contains run time information (status messages) from BAES as it starts up. Once BAES successfully starts, the MAESLOG messages are routed to the MAESLOG DIV data set. It is recommended that you route the information to SYSOUT rather than to a permanent data set.

If this parameter is set to NO, no messages are logged to this data set.

Important! This DD allocation is mandatory. BAES will not start if this DD is missing.

More Information:

Specify MAES Runtime Parameters (see page 193)

//MAESPARM DD DSN=maesparm,DISP=SHR

This PDS data set should contain at least the BAESIMS member. BAES initially reads a member BAESIMS to inquire about the run time parameters.

//MAESLUNM DD DSN=&MAESHLQ...MAESPARM(&LUNAMES),DISP=SHR

The MAESLUNM DD statement specifies a sequential data set that describes the potential session partners. If the DD statement does not exist, MAES cannot run. You must define each possible BAES session partner's characteristics.

The MAESLUNM data set must contain fixed length, 80-byte records without sequence numbers in columns 73 through 80. For example:

```
DCB=(LRECL=80,RECFM=FB,BLKSIZE=3120,DSORG=PS)
```

The MAESLUNM data set is a series of source statements in the form:

```
KEYWORD(PARAMETER) KEYWORD(PARAMETER) -
KEYWORD(PARAMETER)
```

The keyword can start in any column as long as the keyword and its corresponding parameter fit on the same line. Each keyword must have a corresponding parameter. A definition can be continued on a subsequent line by ending the preceding line with a hyphen (-).

The following list describes the supported keywords and parameters for the MAESLUNM data set. The keyword is required unless the word *optional* appears in parentheses next to the keyword.

LUNAME

This MAESLUNM parameter is the LU name of the session partner to BAES. The following list describes the LU names you can specify for the given partner:

MAES

The node name of the partner MAES region

WINCLIENT

Used for Windows SNA clients using independent LU 6.2

MAX (optional)

The parameter specifies the maximum number of sessions that can be acquired to bind this BAES with its session partners. If the session partners specify different maximums, the value used is equal to the lower maximum value. For instance, if a second BAES is bound and specifies a maximum of 5 sessions and the first BAES specifies 10 sessions, only 5 sessions are bound. If MAX is not specified, the originating partner's MAX value is used. If the MAX value is reached, and BAES receives another BIND request, the bind is rejected.

WIN (optional)

The parameter specifies the number of sessions that this BAES owns for initiating transactions with the partner. The actual number of sessions this BAES owns depends on the corresponding specification in the session partner. In general, if the specifications for MAX and WIN differ between two potential partners, the first one initialized determines the number of sessions owned by each. Where the WIN is less than the MAX, at least one session is allocated to the partner. For remote knowledge base partners connecting one BAES to another, WIN must be greater than zero.

ACQUIRE (optional)

This MAESLUNM parameter is either YES or NO. If you specify ACQUIRE(YES), BAES attempts to bind a session with the session partner at initialization. If the bind fails, BAES continues and any binding must be done by the session partner or through the MONITOR knowledge base. If ACQUIRE(NO), BAES leaves the session unbound and expects the session partner to initiate the session later. ACQUIRE(NO) is the default.

MODENAME

This MAESLUNM parameter specifies the entry in the VTAM mode name table that is used by BAES when BAES attempts to establish a VTAM session with its partner.

SYSTYPE

This MAESLUNM parameter generically specifies the session partner's system or program type. The possible values are MAES and WINCLIENT. BAES uses this parameter to establish the correct protocol for binding with its session partner. You must specify this parameter.

For example, in the following definition, the session partner is a MAES region used for BAES:

LUNAME(**MAES**) MODENAME(AIONMODE) SYSTYPE(MAES) ACQUIRE(YES) MAX(10) WIN(0)

Note: To properly connect MAES and BAES regions through VTAM, the MAX and WIN parameters must be coded in the MAES DD MAESLUNM entry (for a BAES session as a MAES session partner).

If a session partner attempts to bind a session with BAES and there is no entry in the MAESLUNM file, the bind is rejected.

Specify BAES Run Time Parameters

This section explains the BAES run time parameters coded in the MAESPARM DD allocation. Each parameter has a default value that takes effect when the parameter is omitted.

ESTAE=[YES|NO]

You can turn off the ESTAE and SPIE functions BAES issues. Turning off ESTAE and SPIE prevent BAES from doing a controlled shutdown if an abend occurs. It is most useful in debugging when a program check has occurred. If ESTAE=YES is specified, the actual abend is masked, making debugging more difficult. If ESTAE=NO, then MAXABEND=1. The default is YES.

STATINT=hhmmssth

The STATINT execution parameter specifies the interval that BAES uses to capture system statistics about BAES and its SL/I data server services. The value is expressed as *hhmmssth*, where *hh* is hours, *mm* is minutes, *ss* is seconds, and *th* is tenths and hundredths of a second.

A value of zero (0) disables the statistics gathering function. The default is 00003000.

ITRACE=nnn

ITRACE defines the number of internal trace entries BAES maintains per VTAM session. The internal trace is used by Aion support for debugging purposes. A value of zero disables the internal trace. The default is 10.

MAESLOG=[YES|NO|LOG_#S]

BAES implements a low-level logging facility that sends execution traces and diagnostic messages to the MAESLOG DD. This log is particularly useful when Aion Support is required to help solve a problem. The default is YES.

- YES enables BAES logging to MAESLOG DD.
- NO disables BAES logging to MAESLOG DD.

If you want to limit the range of error messages produced, you can specify a range of log numbers for the MAESlog to show if they occur. For example, the following command specifies that BAES only write error numbers 1-1000 and 3000-4000 to the MAESlog:

```
MAESLOG=(1-1000,3000-4000)
```

You can change the value using the MODIFY command. However, if you wish to subtract messages numbers from the set to be printed, you must first specify:

```
MAESLOG=NO
```

Then specify all the numbers you wish to be printed. Otherwise, all modifications add to the list.

More Information:

The Multitasking Aion Execution System (see page 171)

MAXABEND=nn

This value indicates the number of BAES abend conditions that will be allowed to occur before BAES terminates. A value greater than one causes BAES to accept that number of internal errors. If the run time parameter ESTAE=NO, then MAXABEND is set to one (1.) The default is 10.

NODENAME=applid

This keyword allows the systems programmer to override the default VTAM application identifier name (APPLID). The default is BAES. You can override the name for the following reasons:

- Installation standards require a name other than BAES.
- The systems programmer wants to run two or more copies of BAES.

Note: The name cannot be longer than eight characters, and must match the entry in the VTAM tables.

PMEMBER=pmember

This parameter specifies the name of the MAESPARM PDS member containing BAES execution parameters. BAES initially reads the BAESIMS member to inquire about its execution (run time) parameters. This entry causes BAES to process another member (PMEMBER) to set BAES execution parameters.

For example, you might have a set of MAESlog specifications that you want to use for debugging at a certain time of day. At the same time you want to change the wait time limit for a consultation to allow for longer periods. Assume you have coded the following entry and put it into a member named OVERNITE in the PDS associated with the MAESPARM DD statement:

```
MAESLOG=(1-3000,5001,6000-9000)
TIME=2000
```

At the appropriate time, the console operator types the following statement:

```
MODIFY jobname,PMEMBER=OVERNITE
```

Assuming the MAESPARM DD statement is present in the BAES JCL, the member OVERNITE is read, interpreted, and new values set. If there is no MAESPARM DD statement, the request fails.

Since PMEMBER can be embedded within other members, you could have a member called HOLIDAY coded as follows:

```
CPUTIME=10.00
PMEMBER=OVERNITE
```

The CPUTIME is set to 10 seconds and the OVERNITE member is read and interpreted.

ROUTCDE=nn

Values can be between 1 and 16, inclusive. Certain actions within BAES produce operating system WRITE TO OPERATOR (WTO) requests. Assume, for example, that the operator modifies the BAES job using the MODIFY command. Completion information is presented using a WTO.

With the ROUTCDE operand, you can direct the output to an appropriate destination. ROUTCDE=11 writes the messages to the job log. This is commonly called a WRITE TO PROGRAMMER.

The value can be between 1 and 16, inclusive. The default is 11. Consult your systems programmer for specific route codes used at your installation.

The format of the operand is as follows:

```
ROUTCDE=nn
```

or

```
ROUTCDE=(nn,nn,...)
```

TIME=nnnn

BAES allows you to set a limit on the amount of time a process waits for input. If this time limit is exceeded, the process is terminated and its resources are returned to BAES. Typically, this parameter may be used if the resources available to BAES are limited, and you want to ensure that processes are not idle for extended periods.

The time is specified as the number of seconds. The default is 900 seconds (15 minutes).

If you wish to disable the wait time function altogether, you can specify:

```
TIME=0
```

If wait time is disabled, BAES will not monitor process wait time, allowing a user to begin a process, and then perform some other activity before returning to the process. You may find this an appropriate setting if any of your applications use the consultation pause function described in the section on Message Switching.

WTOR=[YES|NO]

During initialization, BAES generates the following WTOR system console message:

```
BAES - ENTER 'T' TO TERMINATE Aion IMS CPO-BAES
```

This parameter enables or disables the WTOR message. By specifying WTOR=NO, the WTOR will not be displayed. The default is YES.

Terminate BAES

The BAES job can be terminated by replying to the outstanding WTOR message, or by stopping the BAES job directly. To terminate BAES, do one of the following:

- Terminate BAES from the system console by typing a **T** in response to the WTOR message (issued by BAES during the initialization).
- Enter the following command from the z/OS system console:

  /STOP *baesjobname*

  where *baesjobname* is the name of the BAES job.
- If the WTOR run time parameter is set to NO, purge the BAES job from the active queue.

Appendix A: Accounting Record Format

This appendix describes the format and the content of the accounting record.

ACCTREC

An assembler language DSECT for the accounting record (the ACCTREC member of the <AIONHLQ>.MACLIB library) is included on the MAES distribution tape. The ACCTREC macro automatically generates a DSECT.

- To generate the record format directly into your control section, specify the following:

  ```
  ACCTREC DSECT=NO
  ```

- To use the macro in an assembler program, specify the following:

  ```
  ACCTREC
  ```

In either case, the ACCTREC label is the beginning of the DSECT of the record format. The size of the record format is associated with the label ACCTRECL.

In the following list, unless otherwise indicated, each field is a four-byte binary counter (fullword). All the references to VTAM services also apply to cross-memory services.

Field	Description
ACCTKB	Specifies the 8-character name of the knowledge base that was executed (or *BUILD if the user was editing an application) If applications are chained, this field is the name of the first application only.
ACCTLT	Specifies the 8-character name of the user's terminal
ACCTUSER	Specifies the 8-character user ID This is valid only if the CICS or IMS system is configured to require user signon. If your system does not require user signon and the user is not overriding the ID when the transaction driver is run, this field is filled with blanks.
ACCTRSTO	Specifies the time (in hundredths of a second) when the consultation ended (fullword)
ACCTRSTA	Specifies the time (in hundredths of a second) when the

Field	Description
	consultation started (fullword)
ACCTRTCP	Specifies the amount of CPU time in timer units used by the consultation (fullword)
ACCTRSUS	Specifies the number of times the consultation was suspended This reflects the number of times the consultation was waiting for user input.
ACCTRRUO	Specifies the number of VTAM request units sent to the transaction driver A *request unit* is the smallest unit of transmission under VTAM.
ACCTRRUI	Specifies the number of VTAM request units received by SAES/SADS from the transaction driver.
ACCTRSCO	Specifies the number of screen messages sent This is the number of output messages that were sent to the transaction driver to be displayed on the user terminal.
ACCTRSCI	Specifies the number of user responses sent from the transaction driver to SAES and SADS
ACCTRDAO	Specifies the number of data requests (messages) sent to the transaction driver Each request represents a DL/I, remote IDMS, remote VSAM, or remote DB2 request. The request might be an update request, such as a SELECT statement, or the segment search arguments used to drive the DL/I component of the transaction driver.
ACCTRDAI	Specifies the number of data elements that the transaction driver returned to SAES and SADS This is the number of returned segments, records, or rows. This includes status messages that the transaction driver generated. Error information is typically returned as a status message.
ACCTROMO	Specifies miscellaneous messages sent to the transaction driver This includes such items as status messages, report output, and user data messages.
ACCTROMI	Specifies miscellaneous messages that the transaction driver sent to SAES and SADS This includes user data messages and status messages.
ACCTRBYI	Specifies the number of bytes given to VTAM for

Field	Description
	transmission This does not include VTAM overhead, such as the transmission header.
ACCTRBYO	Specifies the number of bytes returned to SAES and SADS after the VTAM RECEIVE completes This excludes VTAM overhead such as the transmission header.
ACCTRSTE	Specifies the number of states loaded during the execution of the knowledge base
ACCTRRKB	Specifies the number of reads and writes performed by the inference engine, exclusive of VSAM and QSAM I/O
ACCTRKBX	Specifies the number of EXCPs to the KBE data set
ACCTRNPG	Specifies the number of times external programs were loaded during the execution of the knowledge base
ACCTCODE	Specifies the termination code of the consultation or development session If the session terminates abnormally, the abend code is captured in this field. The field is a fullword value with the abend code right-aligned in the field. If the termination code is a system abend, the high-order (left-most) bit of the field is turned on. Otherwise, the abend is a user abend.
ACCTLUNM	Specifies the VTAM LUname where the consultation originated
ACCTCORE	Specifies the amount of virtual memory the consultation used
ACCTDSTR	Specifies the four-byte starting date for the consultation The date is packed decimal digits. The format is 00*yydddf,* where *yy* is the year, *ddd* is the data within the year, and *f* is a sign allowing the data to be treated as a properly packed decimal field.
ACCTDEND	Specifies the four-byte ending date for the consultation The format is the same as ACCTDSTR.
ACCTMAXS	Specifies the maximum amount of memory allowed for the consultation Exceeding the maximum memory amount causes the consultation to terminate.
ACCT24BT	Specifies the amount of memory specifically allocated in a 24-bit address space

Field	Description
	Although all consultations and most of MAES reside above the 16MB, certain I/O functions and their buffers must reside below the 16MB line. This four-byte binary counter indicates the maximum amount of below-the-line memory that was allocated during the consultation.

Appendix B: Messages and Codes

This appendix describes error Messages and Codes for CA Aion BRE.

Interpret Messages and Codes

The MSGID field displays messages in the format xxxnnnnn, where xxx is a letter code, and nnnnn is a log message number. For messages having a value of ADS for xxx, the numeric portion of the message corresponds to the log message of the same number in the appendix "Messages and Codes."

Assume, for example, the MSGID value is:

ADS05002

This indicates it is MAESSECS message #5002, which is described in the appendix "Messages and Codes" as:

5002

MAESSECS

MAESSECS is a subroutine of the VTAM SCIP exit coded into MAES. When MAESSECS receives a BIND request, the requestors LU name is matched against the LU names specified in the MAESLUNM file. This log message indicates that the session is from an IMS-type partner (no parallel session support), and that MAES is about to send an OPNSEC request to the requestor.

The following table shows the type of information provided with each MAES Log message within specific numeric ranges:

Log Message Numbers	Type of Information
1000 - 1999	Task Management
2000 - 2999	VTAM Management
3000 - 3999	Control Block Dump
4000 - 4999	Startup and Termination
5000 - 5999	Partner Management
11000 - 11999	Task Management
12000 - 12999	VTAM Management

Log Message Numbers	Type of Information
13000 - 13999	Control Block Dump
14000 - 14999	Startup and Termination
15000 - 15999	Partner Management

MAES Log Messages (1000 - 19002)

The MAES component creates an output log called MAES log. The MAES log can contain as much or as little information as you desire, according to the values you specify for the MAESlog= execution parameter. The MAES log is designed to track message traffic between MAES and its clients, as well as the status of any consultations within the system.

The MAES log messages (or log messages) have the following format:

```
-- version -- prog date <<msg_no>> time ---
    msg_txt
    [opt_info]
```

Parameter	Description
version	The current version and module level of MAES
prog	The name of the program that created the log message
date	The date the program was assembled
msg_no	A unique log message number
time	The time of day and date when the log message was created
msg_txt	An optional text message
opt_info	Specifically labeled control blocks used by the system (such as RPL or ALOG), or a hexadecimal dump of information needed for debugging

Log messages from 1 to 999 are informational only. Log messages 10000 and higher indicate error conditions that require corrective action.

The following table lists the types of information that each log message contains within the specified log message range:

Log Message Number	Type of Information
1000 - 1999	Task Management
2000 - 2999	VTAM Management

Log Message Number	Type of Information
3000 - 3999	Control Block Dump
4000 - 4999	Startup and Termination
5000 - 5999	Partner Management
11000 - 11999	Task Management
12000 - 12999	VTAM Management
13000 - 13999	Control Block Dump
14000 - 14999	Startup and Termination
15000 - 15999	Partner Management
18000 - 19000	Components

The MAES log messages are listed below in ascending numerical order.

1000

MAESBFPL

The transaction driver passed data to the inference engine in the form of a Fixed Parameter List (FPL). This does not mean that the transaction driver sent any data using FPL. Instead, the normal method of passing data to the attached subtask is in the FPL format. The transaction driver sends data to the inference engine prior to the starting of a new consultation. In the FPL format is the command line, which is passed to the inference engine. This can be used to verify that the correct knowledge base is being started.

1001

MAESETXR

This log message is issued when a subtask ends normally.

1002

MAESNCON

This log message is issued when a consultation is being initialized and the program being attached is not SAES or SADS-that is, the program being attached is an optimized knowledge base. An optimized knowledge base is run as its own program and is not interpreted under SAES and SADS. The name of the program being attached is printed as part of the log message.

1003

MAESNCON

This log message is issued immediately after the ATTACH SVC is issued. The command line passed to the subtask is printed as part of the log message.

1004

MAESNCON

This log message indicates that a consultation is being resumed. This log message is issued immediately after POSTing the subtask. The consultations are managed using WAIT/POST logic.

1005

MAESSETX

The statistics function ended normally and the MAESSTAT data set was closed without any errors.

1008

MAESITAR

The timing analysis function ended after an error occurred while writing to the output data set.

1500

SAESGET

The text "MASTER POSTED" means that MAES posted the daughter task with an error code. Typically, this is done at shutdown when MAES cleans up the current consultations.

1501

SAESGET

A transaction driver sent the CONSULTATION STOP message to force the consultation to end. This can occur when the transaction driver has problems communicating with the DB/DC environment. The error detected in the transaction driver is propagated back to the consultation, causing an abnormal termination.

1502

SAESGET

When the transaction driver is about to send the next prompt or display to the user, the transaction driver tells the consultation under MAES to enter a wait state. This log message is issued prior to the subtask relinquishing control to MAES.

1503

SAESGET

RWAIT POSTED is issued under the following conditions:

- The transaction driver is resuming a suspended consultation.
- The amount of time that the consultation was waiting for the transaction driver to resume processing was greater than that specified on the TIME= execute parameter.
- MAES posted the suspended consultation to force termination due to a normal shutdown, or other abnormal conditions that were flagged by other MAES log messages.

1504

SAESSEND

This log message is produced when MAES posts the consultation. Typically, the consultation terminates normally.

1505

SAESSTIN

When this log message is issued at consultation initialization, it indicates that SAES, SADS, or an optimized knowledge base was entered properly. If this log message is not issued, the system configuration has significant problems.

1506

SAESPGOT

Paging was forced for this consultation, overriding normal demand paging.

2000

MAES

This log message is issued just prior to the VTAM OPEN call which MAES uses to establish a connection to VTAM for subsequent activities. Any errors in the OPEN call are detected after this log message is issued.

2001

MAESCICS

MAESCICS processes service manager data messages. This status message traces the request unit received on an LU6.2 service manager session.

2002

MAESCICS

This log message notes that a VTAM LUSTAT protocol message was received on an LU6.2 service manager session.

2003

MAESCICS

This log message shows an LU6.2 change number of sessions protocol message that was received on the service manager session.

2004

MAESCICS

This log message displays a response to an LU6.2 exchange log names protocol message only if the Exchange Log Names (XLN) message sent from the session partner was sent on the service manager session.

2005

MAESCNOS

A change number of sessions (CNOS) was sent to the session partner. The message is either an initiated CNOS by MAES or a reply to the CNOS received by MAES.

2006

MAESCSUB

During deactivation of an APPC partner session, MAES cleans up internal control blocks used to manage the traffic sessions. This log message indicates that the cleanup routine was entered.

2007

MAESETXR

This log message indicates that MAESETXR determined that the RPL is still in use. MAESETXR cancels the VTAM RPL request and reuses the RPL. When MAESETXR terminates a subtask, MAESETXR uses the VTAM RPL that belongs to that subtask. If the subtask ended abnormally, the RPL might already be in use for traffic between the subtask and the transaction driver.

2008

MAESLUNM

This log message contains the text of any error condition found in MAESLUNM.

2009

MAESLGON

MAES received a LOGON request. This log message is issued at entry to MAESLGON, but before MAES determines if the request is from a valid session partner.

2010

MAESLGON

In response to a LOGON request, MAES issues a VTAM OPNDST call to the requester. This MAES log message shows the VTAM RPL and node initialization block (NIB) used for the successful OPNDST.

2011

MAESLUNM

The value specified for WIN was not a valid number.

2012

MAESLUNM

The value specified for MAX was not a valid number.

2013

MAESNSEX

In certain VTAM error situations (typically when a session is being unbound), the network services exit is given control. This log message reflects that fact and also logs the header information that is used to determine the reason for the error condition.

2014

MAESRCVE

After an outstanding VTAM RECEIVE call is completed, another RECEIVE call is issued. This log message indicates that the RECEIVE call can be completed with a message from any session bound to MAES which is not already in a specific mode with another MAES component.

2015

MAESLUNM

The value specified for RU was not a valid number.

2016

MAESLUNM

The value of RU is either less than 256 or not a power of 2 (256, 512, 1024, and so on).

2017

MAESSBND

MAESSBND constructs the request unit for binding to another session. This log message shows the contents of the BIND image.

2018

MAESSCIP

A VTAM SCIP EXIT is given control. This log message indicates that one of the following conditions occurred:

- Request Recovery (RQR VTAM message)
- Set and TEST Sequence Numbers (STSN VTAM message)
- Bind request
- Session unbind
- Clear

MAESSCIP determines which condition occurred and acts accordingly.

2019

MAESSCIP

The MAES SCIP exit receives an UNBIND request. The reason code for the UNBIND request is logged with this log message. Typically, an UNBIND request occurs when the partner session terminates.

2020

MAESSECS

The SCIP VTAM exit routine received a BIND request. These requests are routed to MAESSECS. This log message contains the BIND parameters.

2021

MAESSECS

If a BIND request is accepted, MAES issues an OPNSEC call. This log message indicates that the OPNSEC call was issued. MAES checks to make sure that the OPNSEC call was successful.

2022

MAESSERV

Either the MONITOR knowledge base or MAES shutdown processing initiated a service request. This log message indicates that an active service manager session is being unbound by doing a VTAM CLSDST on the session. The session is owned by MAES.

2023

MAESSERV

As a result of a MONITOR request or shutdown processing, MAES terminates a traffic session with a session partner in a parallel sessions connection. This log message indicates that MAES issues a VTAM TERMSESS on the session.

2024

MAESSERV

As a result of a MONITOR request or shutdown processing, MAES terminates a service manager session with a session partner in a parallel sessions connection. This log message indicates that MAES issues a VTAM TERMSESS on the session. The session partner owns the session.

2025

MAESSMGE

This log message records data received on a service manager session that is connected to a session partner in a parallel sessions connection. This MAES owns the session and, typically, reflects responses to MAES-initiated actions such as an exchange of log names (XLN).

2026

MAESSMGE

A message was received from a session response that required a VTAM response. This log message records that a response was sent.

2027

MAESSMGE

In response to a change number of sessions (CNOS) request sent on the service manager session owned by this MAES, the session partner sends the negotiated reply. MAES accepts the reply and logs the CNOS reply.

2028

MAESSMGR

When a MAES binds with a session partner in a parallel sessions connection, MAES binds a service manager session consistent with LU6.2 protocol. This log message is issued after the OPNDST call is executed and before the return code is examined.

2029

MAESSUBS

Prior to acquiring (OPNDST) a sub-session with a session partner in a parallel sessions connection, MAESSUBS logs the VTAM node initialization block (NIB) for informational purposes.

2030

MAESSYNA

A VTAM error occurred. However, the error code indicates that the request can be retried. This log message is issued immediately before the retry attempt.

2031

MAESRECP

This log message is issued when MAESWALL is posted from its wait state because MAES received a VTAM LU6.2 LUSTAT message. The VTAM LU6.2 LUSTAT message is written to MAES log, and then MAES waits for more data.

2032

MAESRECP

MAES received a message that required a VTAM response. The point at which the message was received does not affect any current consultations. MAESERR is called to send the response.

2033

MAESRECP

MAES received a data message. This log message records the VTAM RPL as well as the content of the request unit.

2034

MAESXLN

MAES originates an exchange log names (XLN) request on the MAES-owned service manager session. This is a normal part of LU6.2 protocol.

2035

MAESXLN

MAES received an exchange log names (XLN) request from a session partner in a parallel sessions connection. Typically, this request requires a VTAM response. This log message records that MAES is responding to an XLN initiated by the session partner.

2036

MAESXLN

This log message is issued after an exchange log names request was sent successfully to a partner session.

2038

MAESAUTO

MAES is not connected to VTAM, so a request for automatic acquisition of a session partner was canceled.

2111

MAESLUS

This log message indicates that MAES is sending a VTAM protocol message (LUSTAT), which verifies that a session has been properly established and sets the traffic direction. In the current implementation, the message is not a proper LUSTAT. Instead, the message is a zero length data message with the appropriate protocol switches set.

2112

MAESLUS

This log message is issued after control is returned to MAES following a SEND of an LUSTAT message.

2116

MAESRRN

MAESRRN responds to an FMH7 from a session partner in a parallel sessions connection. An FMH7 typically indicates an abend condition that requires a response. MAESRRN sends the response.

2125

MAESSMGE

MAES attempted to use a service manager session, but the service manager session had already been unbound.

2135

MAESRECP

This log message shows the user's input message and the VTAM RPL.

2208

MAESETXR

This log message indicates a change in the VTAM RPL status for a terminating consultation. If the associated transaction driver is an LU6.2 (with parallel sessions) partner, the RPL associated with the terminating subtask is reset from SPECIFIC to ANY mode.

2500

SAESGET

If the consultation is waiting for data from the transaction driver, this log message is issued. This log message is issued just after a VTAM RECEIVE request is issued but before the WAIT is issued.

2501

SAESGET

This log message is issued when a message is received from the transaction driver. The VTAM request unit is recorded in the MAES log.

2502

SAESSEND

This log message shows the contents of the VTAM RPL request prior to the actual VTAM SEND. This log message verifies the VTAM protocol of the outgoing request unit.

2503

SAESSEND

This log message shows the request unit that was sent as well as the contents of the VTAM RPL and the AESCB. At this point, VTAM accepted the message, and as long as no error indicators are in the RPL, the request unit was sent to the session partner.

2504

SAESSEND

This log message confirms that VTAM returned control to the SAESSEND program, and that the message was accepted without errors.

2505

SAES62

An LU6.2 CONNECT request was issued for a secondary LU6.2 session.

2506

SAES62

The VTAM FMH5 request unit is logged prior to sending it through a secondary LU6.2 session.

2507

SAES62

An LU6.2 SEND request was issued for a secondary LU6.2 session.

2508

SAES62

An LU6.2 RECEIVE or WAIT request was issued for a secondary LU6.2 session.

2509

SAES62

An LU6.2 FREE request was issued for a secondary LU6.2 session.

2510

SAESGET

This log message shows the entire input message from the transaction driver.

2512

SAESGET

A conditional end bracket (CEB) was received from the transaction driver. The VTAM session was reset.

3000

MAESNCON

This log message is issued immediately after log message 1004. This log message is a dump of internal control blocks which technical support can use for debugging task management errors.

3002

MAESWALL

The control blocks were dumped after MAES log message 4023. Log message 4023 is still issued, but the control blocks are not recorded.

3003

MAESWALL

The control blocks were dumped after MAES log message 4033. Log message 4033 is still issued, but the control blocks are not recorded.

3501

SAES62

The AESCB control block is dumped after a VTAM SEND request was issued on a secondary LU6.2 session.

3502

SAES62

The AESCB control block is dumped after a VTAM RECEIVE request was issued on a secondary LU6.2 session.

3503

SAES62

The AESCB control block is dumped after an LU6.2 RECEIVE or WAIT request was issued on a secondary LU6.2 session.

3510

SAESSEND

This log message shows the output message (in hexadecimal) to the user.

4000

MAESINIT

This log message indicates that MAES is active. This log message includes the version number of MAES.

4001

MAES

MAES attempts to open the MAESPARM DD statement. MAESPARM is associated with a partitioned data set containing execution parameter information. If no MAESPARM DD statement exists, all execution parameters are either in the EXEC statement or are dynamically entered using the operator MODIFY command.

4002

MAES

This log message indicates that MAES accepts BIND requests from any partner sessions. This log message is issued after the VTAM SETLOGON request has completed, but before MAES initializes its ESPIE and ESTAE.

4003

MAESAALG

This log message records the building of each ALOG control block. As part of the initialization process for MAES, a new control block is built for each potential session partner.

4004

MAESACCT

MAES opened the MAESACCT data set successfully, which activates normal MAES accounting.

4005

MAESALRU

This log message records request unit memory allocation and request unit size. As part of the initialization process, MAES allocates memory for request units.

4006

MAESAUTO

As part of startup or by virtue of a MONITOR request to acquire a partner, MAES automatically acquires sessions with a session partner. This log message is issued immediately after the VTAM REQSESS request.

4007

MAESBECB

This log message is issued after storage was allocated for the list of event control blocks (ECBs). MAES maintains a list of ECBs.

4008

MAESBRPL

During initialization, MAES constructs VTAM control blocks for all potential consultations. This log message indicates that the VTAM RPL request was built.

4009

MAESDBUG

During initialization, MAES attempts to load the MAESDBUG program that represents an optimized version of an Aion-supplied knowledge base (debug knowledge base) to analyze problems. This log message indicates that the module was found and that the debug knowledge base was activated.

4011

MAESLOGM

MAES can dynamically link to certain external programs (such as the debug knowledge base). The external programs can add log messages to the MAESLOG data set under this ID. These log messages can be informational or reflect error conditions. For example, execution parameter processing errors are recorded in MAESLOG. MAESPARM and PMEMBER processing messages are also associated with log message 4011.

4012

MAESLUNM

This log message indicates that the MAESLUNM analysis function was entered.

4013

MAESLUNM

This log message is issued when a parsing error is found in the MAESLUNM entry. This log message describes the problem. Correct the error and resubmit the MAES JCL.

4014

MAESLUNM

This log message shows the record that is read from the MAESLUNM file before any processing is done.

4015

MAESPARM

This log message has no text. This log message precedes the remaining log messages that are issued when the execute parameters are parsed and analyzed.

4016

MAESPRLD

This log message indicates that the preload function was invoked.

4017

MAESPRLD

This log message shows the name of the program that was successfully loaded prior to running any consultations.

4018

MAESPRLD

This log message indicates that the MAESPRLD DD statement is missing or the data set is empty.

4019

MAESSECS

If a service manager session with the session partner was bound, MAES initiates a VTAM RECEIVE request to ensure that any messages sent by the session partner are processed.

4020

MAESSERV

Certain functions within the system must be performed under control of the MAES task as a means of enforcing serialization. This log message indicates that a system service request was received from a subtask. The nature of the request is explained in subsequent log messages.

4021

MAESSHUT

During shutdown, this log message is the last to appear in the MAES log.

4022

MAESSHUT

MAES can recover from abends within the MAES task. The MAES end of the task exit routine handles subtask abends. ESPIE and ESTAE routines handle MAES abends. If MAESSHUT gains control due to an abnormal condition, it tolerates a certain number of abends before terminating MAES. If that limit is not exceeded, MAESSHUT returns control to the mainline of MAES.

Since this error represents an abend within the MAES task, review the problem and submit your findings to technical support for resolution, where appropriate.

4023

MAESWALL

This log message indicates that the MAES mother task (represented by the MAES program) entered a wait state. This is normal and indicates that no VTAM activity exists, or the active consultations are running or waiting for input.

4024

MAESWALL

During MAES processing, the ECB list that MAES uses needs to be rebuilt periodically. For example, the ECB list needs to be rebuilt when a new session partner is bound, and MAES needs to wait on the new session partner's VTAM session. At that point, the executing program (actually a VTAM EXIT routine) posts the main task out of the wait state so it can rebuild the ECB list.

4025

MAESWALL

This log message is issued when an internal service request is made. MAES responds to a number of asynchronous service requests, which can be external or internal. When a service request must be performed by the mother task, an internal service request is initiated. Usually, the monitor knowledge base initiates an internal service request. For example, a service request to acquire a session partner must be done by the mother task, so an internal service request is issued.

4026

MAESRCVE

This log message indicates that a message was received from the session partner without any error condition set.

4027

MAESRECP

VTAM has a protocol message that is called a CHASE. MAESWALL is designed to respond to a VTAM CHASE. This log message indicates that MAESWALL received a CHASE and is ready to respond.

4028

MAESRECP

Certain LU6.2 messages carry a VTAM format header (FMH). This log message indicates that MAES recognized the FMH presence and deleted it, leaving only the application information.

4029

MAESRECP

Part of the LU6.2 protocol involves a data message that is used to carry switches that control session ownership. This log message indicates that the data message was received and is only used to manage internal state conditions.

4030

MAESWALL

MAES terminates if the Write to Operator with Reply (WTOR) was answered properly. This log message indicates that the WTOR was answered, but the answer was not verified.

4031

MAESOPER

MAES responds to an operator STOP command (STOP *jobname*). This log message indicates that an operator STOP command was recognized. MAES terminates.

4032

MAESWALL

MAES responds to an operator MODIFY command (MODIFY *jobname*,*parameters*). This log message indicates that an operator MODIFY command was recognized. MAES passes the parameter information to MAESPARM for processing and continues.

4033

MAESWALL

MAES waits on certain events to complete. One event is a consultation being suspended. In this case, the VTAM session is released back to the pool and MAES is posted, indicating that the subtask is suspended.

4035

MAESINIT

This log message is issued when MAES runs as an authorized program. Authorized path VTAM is used for all synchronous VTAM requests.

4036

MAESINIT

This log message is issued after the ADSAPI module is loaded and called.

4037

MAESPGSI

This log message contains messages issued by the Aion-supplied program security initialization module (MAESPGSI).

4038

MAESINIT

This log message displays the return code when the program security initialization program has been linked.

4039

MAESISTT

Statistics gathering is enabled.

4040

MAESISTT

The statistics gathering function is disabled. The DD statement for MAESSTAT was not found.

4041

MAESSWAP

This log message is issued during startup when the MAES address space is swappable. The MAES address space should be nonswappable otherwise adequate performance will not be assured. This message also prints out the default user ID that is used for RACF and DB2 authorization checking.

4042

MAESINIT

The SAESEP interface module loaded successfully.

4043

MAESINIT

The MAESHPO data set was opened successfully.

4044

MAESINIT

The MAESHPO DD statement is missing.

4047

MAESITAR

The timing analysis function was started.

4048

MAESITAR

The timing analysis function was not started. The output data set (the MAESMTAR DD statement) could not be opened.

4050

MAESINIT

The SESSLOG data set was opened successfully. The monitor knowledge base can capture message activity.

4051

MAESINIT

The SESSLOG data set could not be opened. The MONITOR knowledge base cannot capture message activity.

4060

MAESAPPS

This log message indicates that the pre-initialize module (MAESAPPS) was invoked.

4061

MAESAPPS

This log message shows the name of an application that was successfully pre-initialized prior to running any consultations.

4065

MAESAPRM

This log message indicates that the application LE run-time parameter module (MAESAPRM) was invoked.

4066

MAESAPRM

This log message indicates that the MAESAPRM DD statement is missing or the dataset is empty.

4067

MAESAPRM

This log message indicates that LE run-time parameters were extracted from the MAESAPRM dataset for the named application.

4070

MAESERR

This log message shows the last message that the consultation sent to the transaction driver.

4089

MAESOPER

The MAESXSUB load module could not be brought into memory. The MAESLOG data set cannot be used.

4098

MAESINIT

The MAESXSUB load module could not be brought into memory. The MAESLOG data set cannot be used.

4118

MAESRECP

MAESRACF was called and returned a nonzero value in register 15.

4129

MAESRECP

A VTAM message with a length equal to zero was received and ignored.

4141

MAESSWAP

The MAES address space is nonswappable.

4142

MAESPRLD

The optimized module is eligible for forced paging.

4409

SAESTERM

This log message shows user output data in hexadecimal form.

4410

SAESTERM

This log message shows the MST (an internal control block) of the ending consultation.

4500

SAESSTIN

This log message is issued each time a consultation is initialized. This log message contains version, modification, and date information that can be used to verify the SAES or SADS level that you are running.

4503

SAESRUID

This log message shows the PDBs (Parameter Descriptor Blocks) of the input Fixed Parameter List.

4504

SAESRUID

This log message records the parameter list specifying where input and output data is to be placed. Technical support can use this information for debugging.

4505

SAESRUID

This log message records the input data area for FPL data.

4506

SAESRUID

This log message indicates that all of the information contained in the last request unit was received from the transaction driver during FPL input retrieval.

4507

SAESRUID

This log message indicates that the transaction driver is not sending any FPL input data. However, there might be FPL output data.

4508

SAESSTAT

As part of normal protocol, SAES/SADS sends status messages to the transaction driver. This log message indicates that the subroutine is about to send the status message.

4509

SAESTERM

As part of FPL processing, output data is returned to the transaction driver at knowledge base termination. This log message contains the data that is to be returned to the session partner.

4510

SAES62

As part of the underlying secondary LU6.2 session support for remote knowledge bases, this log message is issued to indicate that a VTAM RECEIVE request is about to be issued.

4900

MAESXMCH

During shutdown, MAESXMCH found that there are still active cross-memory services sessions.

5000

MAESLGON

If a potential session partner attempts to logon to MAES, MAESLGN is given control. If appropriate, MAES attempts an OPNDST request. This log message indicates that the session partner is valid and the OPNDST request is about to be issued.

5002

MAESSECS

MAESSECS is a subroutine of the VTAM SCIP exit coded into MAES. When MAESSECS receives a BIND request, the requester's LU name is matched against the LU names specified in the MAESLUNM file. This log message indicates that the session is from an IMS-type partner (no parallel session support) and that MAES is about to send an OPNSEC request to the requester.

5003

MAESSECS

MAESSECS is a VTAM SCIP exit subroutine that handles BIND requests. In this case, a BIND request was received from a session partner that is defined as supporting parallel sessions (APPC). Also, the mode name is SNASVCMG. These two conditions indicate that the requester is attempting to bind its service manager session with MAES.

MAES is about to issue the VTAM OPNSEC request to complete the bind.

5004

MAESSERV

MAES received a request to deactivate a session partner. That means breaking any sessions bound with that session partner and marking it as unavailable to accept BIND requests. The request was either from the MONITOR knowledge base or a result of normal shutdown.

5005

MAESSERV

As part of session partner deactivation, MAES (where the session partner supports parallel sessions) unbinds (using a CLSDST call) the MAES-owned service manager session. This log message records this event.

5006

MAESSERV

MAES received a request from the MONITOR knowledge base to activate an inactive session partner. Unless a session partner is active, BIND requests are rejected.

5007

MAESSERV

MAES received a request from the MONITOR to acquire sessions with the session partner. This is applicable only to session partners that support acquisition by MAES. This results in binding VTAM sessions with the session partner.

5008

MAESSERV

As part of remote knowledge base processing, mainframe-to-mainframe, traffic sessions to the session partner are allocated from the pool of sessions owned by this MAES. This log message records that a request for a session was received.

5009

MAESSMGR

As part of the LU6.2 protocol in which parallel sessions are involved, each side of the connection owns a service manager session. This log message indicates that MAES is about to issue an OPNDST to establish a service manager session with XXXXXXX.

5010

MAESSTAB

Sessions are bound with a session partner that was previously bound with MAES. This log message indicates that an internal control block is freed prior to allocating a new one.

5011

MAESSTAB

Sessions are bound with a session partner in a parallel sessions connection. This log message indicates that internal control blocks are allocated to support the traffic sessions.

5012

MAESSUBS

As part of the process to establish traffic sessions with a session partner in a parallel sessions connection, MAES initiates BIND requests to the traffic sessions MAES will own. This log message indicates that an attempt to bind a traffic session using an OPNDST call was made.

5013

MAESSUBS

This log message is issued after MAES completed acquiring all the traffic sessions it should own for a particular session partner.

5014

MAESSERV

The data sets associated with the MAESHPO DD statement can be closed and reopened using the MONITOR knowledge base. This log message is issued if the action is successful.

5015

MAESSERV

The MONITOR knowledge base can be used to suspend scheduling of an optimized knowledge base. This log message indicates which knowledge base is being quiesced.

5016

MAESQPGM

An internal queue of optimized knowledge bases, which are preloaded, quiesced, or copied (new), is built. After MAES initialization, this log message is issued when a MONITOR knowledge base action causes a new program queue control block to be built.

5017

MAESSERV

This log message indicates that a previously quiesced, optimized knowledge base is being activated and made available for scheduling.

5018

MAESSQ

This log message is issued when a user of the MONITOR knowledge base requested that an optimized knowledge base be refreshed (commonly called newcopy). The storage shown in this log message is the name of the optimized knowledge base being reloaded.

5019

MAESSERV

The MONITOR knowledge base can be used to change the dispatching priority of an optimized knowledge base's future schedulings. This log message indicates the name of the optimized knowledge base and its new priority.

5023

MAESSERV

The monitor knowledge base is preloading a program.

5106

MAESSERV

This log message is issued when the MONITOR knowledge base is activating a session partner.

5111

MAESSTAB

As a result of a change number of sessions (CNOS), a new subsession table has been allocated.

6000

MAESMALL

This log message indicates that the cross-memory services environment was created under MAES.

6001

XAESNCON

A cross-memory bind was received and accepted by a transaction driver.

6002

XAESNCON

MAES received a cross-memory message. The cross-memory message is a control message rather than a traffic message for the consultation.

6003

SAESXMSE

A cross-memory message is about to be transferred to the transaction driver.

6004

XAESCLEN

A cross-memory session is being terminated.

6005

SAESXMGE

A cross-memory message is about to be retrieved from the transaction driver.

6007

XAESNCON

This log message indicates that a protocol message was sent after a cross-memory bind.

6008

XAESERN

A cross-memory status message is about to be sent to the transaction driver.

6009

XAESNCON

A cross-memory event completed successfully.

6010

SAESXMSE

A cross-memory send completed successfully.

6011

MAESMALL

To assist in debugging, this log message traces the Aion-written SVC.

6016

XAESNCON

This log message shows the transaction driver version.

6017

XAESNCON

The transaction driver version does not match the MAES version. The consultation request is rejected.

7010

SAESAPI

This log message shows the contents of the buffer that the asynchronous API uses when a message is sent to the client.

7020

SAESAPI

This log message indicates that the asynchronous API requests a message from the API client.

7030

SAESAPI

This log message contains the return code from an API request.

7040

SAESAPI

This log message contains the asynchronous API message received from the client.

11000

MAESBFPL

When a consultation is initialized, storage is allocated to hold the parameter list that is passed to the subtask. This includes any FPL information and the command line. If the MAES region is too small, there might not be enough storage available for the parameter list. This log message records the problem before returning an error code to the caller. The consultation request is rejected.

Increase the MAES region size to provide enough storage for the parameter list.

11001

MAESETXR

When a consultation terminates, the address of the terminating task control block (TCB) is used as an argument into a chain of internal control blocks. If none of the control blocks has a matching TCB address, this log message is issued.

If this occurs, MAES has an internal error. This error should be reported to technical support. Although the subtask terminates normally, the terminal that was running the consultation will probably hang. MAESETXR needs to locate the control block to properly terminate the connection with the corresponding transaction driver. Moreover, the subtask that dynamically allocated the data sets did not properly free the data sets. The data sets remain allocated.

11002

MAESETXR

The MAES end-of-task exit routine detects consultation abends. System and user abends are formatted into a message indicating the nature of the abend. For user abends, the decimal abend code typically matches a message number in the consultation's MAES system log. Typically, user abends result from consultation errors. You will probably need a system log to resolve user abends

For system abends, you need to convert the decimal abend code into a hexadecimal number to analyze the problem. For instance, a protection exception appears as system abend 0196 in this log message, but is normally traced as system abend OC4 (in hexadecimal). To resolve system abends you need to allocate a SYSUDUMP DD statement to the MAES job.

11003

MAESNCON

This log message indicates that an existing consultation was terminated in order to start a new consultation. MAES uses a combination of the session partner's LU name (for parallel session connections) and the user's terminal name to identify a consultation. Under certain abnormal situations, even though an existing consultation was suspended, the transaction driver attempts to initialize a new consultation with the same identifier (LU name and terminal name). MAES automatically terminates the old consultation before initializing the new consultation.

11004

MAESNCON

This log message is issued when the maximum number of consultations that can run concurrently is exceeded. The MAXTASKS execution parameter sets the maximum number of consultations that can run concurrently.

In the MAXTASKS execution parameter, increase the maximum number of consultations that can run concurrently. You might also need to increase the amount of memory available to MAES.

11005

MAESNCON

This log message is issued when MAES receives a consultation resume request and the consultation is not active. This log message includes the consultation identifier (typically the terminal ID). This condition can occur if the consultation timed out or was otherwise terminated while the end user was answering a prompt. MAESNCON manages resuming current consultations or starting new consultations.

11006

MAESSHUT

As part of shutdown processing, MAES abnormally terminates any consultations that are in process. This log message identifies each consultation that terminated. Each consultation is identified by its consultation ID.

11007

MAESRECP

MAES was posted from its WAIT state. MAESWALL receives a message associated with a specific consultation, but MAESWALL cannot locate the consultation's internal control block. This error should be reported to technical support.

11008

MAESRECP

This log message indicates that the consultation associated with the session receiving the bracket initiate stop (BIS) was posted to terminate. The VTAM protocol BIS message indicates that the receiver is to shutdown any activities on that session.

11009

MAESSETX

This log message shows the abend code for the statistics routine that has abended.

11010

MAESITAR

This log message contains abend information for the error that occurred while writing to the timing analysis data set. Timing analysis ends.

11011

MAESNCON

This log message is issued when #ACQUIRE command has been received, but the MST pre-allocated by MAESSECS could not be located. The CID being used to identify the conversation is in the SESSID field in the SESS control block accompanying this message.

11500

SAESGET

This log message is issued when the consultation was in a WAIT state longer than the limit specified in the TIME= execution parameter. The consultation abnormally terminates.

11501

SAESSEND

MAES posted the subtask with a code indicating that the consultation should be shut down. Two conditions can cause this to occur:

- A consultation cancellation request was received from the MONITOR knowledge base.
- A new consultation is being started with the same ID as the current one.

11502

SAESSEND

MAES can post SAESSEND when normal shutdown is in progress. This log message indicates that a consultation was in progress when normal shutdown began. Other log messages that indicate the cause of the shutdown are issued.

11503

SAESSEND

This log message indicates that MAES posted the subtask, but the subtask could not recognize the code it was given. MAES can post SAESSEND for consultation cancellation and system shutdown.

11504

SAESSTIN

This log message indicates that the consultation used too much CPU time to reply to the user's last input. The consultation abends with user abends 4000.

11505

SAESSTOP

This log message indicates a logical error in the consultation that can be resolved using other messages in the MAES log. While processing a consultation, the low-level VTAM routines within SAES or SADS either timed out or received a stop consultation message from the transaction driver. After this log message is written, SAES abends.

12000

MAES

MAES attempts to establish a connection to VTAM by opening an Access Method Control Block (ACB) using the nodename provided in the NODENAME execute parameter. If the OPEN request fails, MAES formats the VTAM error code. The error code can be located in the *IBM VTAM Programming* manual. If the debug knowledge base is available, more detailed information about the error and a suggested resolution to the problem is written to the MAES log.

12001

MAES

After opening its VTAM ACB, MAES enables itself to accept BIND requests. If the VTAM request (SETLOGON) fails, MAES logs the error code and terminates. If the debug knowledge base is available, additional detailed information is written to the MAES log.

12002

MAESAUTO

MAES had an error issuing the VTAM REQSESS request when MAES attempted to automatically bind with a potential session partner. MAESSYNA logs more information.

12003

MAESBRPL

When setting up potential consultation subtasks, MAES uses VTAM services to generate control blocks. This log message occurs when the NIBGEN fails. The error code is included in the log message, and can be used in conjunction with the *IBM VTAM Programming* manual to help resolve the problem.

12004

MAESBRPL

When setting up potential consultation subtasks, MAES generates control blocks using VTAM services. This log message occurs when the RPLGEN fails. The error code is included in this log message and can be used in conjunction with the *IBM VTAM Programming* manual to help resolve the problem.

12005

MAESCICS

The MAESCICS routine asynchronously processes traffic from a parallel-sessions-connected partner's service manager session. An error condition was detected after the VTAM RECEIVE and the VTAM CHECK request were completed. This situation can occur when the session partner abnormally terminates or is canceled without cleaning up properly. If this is not the case, contact technical support and have the MAES log available. A log message from MAESSYNA further analyzes the error.

12006

MAESCICS

MAES attempts to respond to an exchange log names request from a session partner in a parallel sessions connection. However, VTAM detected an error. This message is accompanied by a log message from MAESSYNA that further analyzes the problem.

12007

MAESCNOS

MAES either initiates or responds to a session partner's change number of sessions request. VTAM detects an error on the request. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12008

MAESERR

MAES attempts to send a status message to the transaction driver. VTAM detects an error. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12009

MAESLGON

MAESLGON issues an OPNDST call in response to a CINIT from a potential session partner. VTAM detects an error condition. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12010

MAESLUS

MAES attempts to send a VTAM LUSTAT message to a session partner. VTAM detects an error condition. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12011

MAESNSEX

When the VTAM network services exit gets control, a header code is used to determine the service that is required. This message formats the header code for debugging purposes.

12013

MAESRCVE

MAESRCVE issued a VTAM RECEIVE request against a session and the request failed. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12014

MAESRCV1

MAES issued a VTAM RECEIVE request against its service manager session. VTAM detected an error condition. A log message from MAESSYNA, which precedes this log message, further analyzes the problem.

12015

MAESSCIP

The VTAM SCIP exit handles a set of conditions such as BIND and UNBIND. This log message indicates that the MAESSCIP program could not determine what action was requested. Report this situation to technical support. Include a copy of the MAES log in your report.

12016

MAESSECS

MAES issues an OPNSEC request in response to a session partner's OPNDST request. If the VTAM request fails, this message is issued as an analysis by MAESSYNA. The error should be reported to technical support.

12017

MAESSECS

After MAES receives a BIND request, it verifies the following:

- The LU name is in the MAESLUNM data set.
- The potential session partner supports multiple request unit output.
- The node is active (has not been deactivated using the MONITOR knowledge base).
- The log mode name is present in the BIND parameters.

If any of these conditions is not valid, MAES issues a SESSIONC request to reject the BIND request.

12018

MAESSMGE

A VTAM RECEIVE request issued against the MAES service manager session that is bound with a session partner, was posted with an error code. A log message from MAESSYNA, which precedes this log message, has more information about the error condition.

12019

MAESSMGR

MAES attempts to bind a service manager session with a session partner (*xxxxxxxxx*). The OPNDST request fails. Preceding log messages from MAESSYNA provide a further analysis of the problem.

12020

MAESSUBS

MAES attempts to acquire a traffic session. The OPNDST request fails. Preceding log messages from MAESSYNA further analyze the problem.

12021

MAESSYNA

MAESSYNA manages cancellations of outstanding VTAM requests. If MAES is waiting to receive a message from its session partner, and the session partner is canceled, or terminates without replying, VTAM automatically posts the outstanding RECEIVE request with an error condition. This log message notes that a RECEIVE request was canceled. This is typical for sessions with a single session connection, but is not typical for sessions with a parallel session connection.

12022

MAESSYNA

All VTAM request errors, other than OPEN and CLOSE requests, force the SYNAD exit to be executed. MAES logs the return code and feedback information in the VTAM RPL with this message. If the debug knowledge base is active, detailed information is written to MAES log. This can help in analyzing the problem.

12023

MAESSYNA

Certain VTAM errors can be retried. If the MAES VTAM SYNAD exit determines that a retry is appropriate, a VTAM EXECRPL is executed. This message indicates that the request was rejected. The RPL is logged so that technical support can debug the problem.

12024

MAESRECP

MAES received a reached recovery node (RRN) VTAM protocol message from the session partner. MAES calls MAESRRN to respond appropriately.

12025

MAESXLN

By returning its own XLN message, MAES responds to an exchange log names (XLN) request from a session partner. If the VTAM request fails, MAESSYNA issues an error message. The code that issued the SEND request issues this log message.

12026

MAESSECS

MAES received a BIND request, but the name of the logical unit attempting the BIND was not found in the MAESLUNM data set. Determine if the attempt to bind was valid. If the BIND request was valid, correct the MAESLUNM data set before resubmitting the request to MAES. You can also add a new potential partner session using the MONITOR knowledge base if you are unable to terminate MAES to make the correction.

12027

MAESSECS

MAES received a BIND request and at least one matching model LU entry was found in the MAESLUNM file, but all corresponding ALOG entries were already in use. Increase the number of model LU entries in the MAESLUNM file so that it corresponds to the number of concurrent client requests using the model LU name.

12118

MAESSECS

The session control blocks are dumped after a BIND fails. A log message header (with the same log message number) separates each session control block dump.

12140

MAESERR

A format header type 7 (FMH7) was sent to the transaction driver.

12145

MAESERR

A status message was sent to the transaction driver. This log message contains the status message in hexadecimal form.

12240

MAESRCV1

A VTAM RECEIVE call was made to a service manager session.

12250

MAESRESP

A VTAM response was sent to the transaction driver.

12500

SAESGET

This log message is issued when a VTAM protocol message SEND fails. For instance, SAESGET detects a VTAM SHUTD request and attempts to respond. If the response fails, log message 12500 is issued.

12501

SAESRUID

This log message is issued when the low-level VTAM routines detect an error. Analyze the error situation using the log message that SAESGET writes to the MAES log prior to this log message. If a true VTAM error has occurred (not a logical error managing the message), MAESSYNA writes a log message to the MAES log.

12502

SAESRUID

When the low-level VTAM receive routine is called, the expected size of the user input data is passed. This log message indicates that the request unit size received by SAESGET was greater than the size that SAESRUID expected. This is a programming error in ADSFPL, AAESFPL, or DCFPL. The amount of data actually sent must be the same amount that SAESRUID expects.

12620

SAESSEND

The calling function attempted to receive a message. However, the session was in send mode, so the request was rejected with an error.

13000

MAESNCON

When MAES is unable to resume a consultation because the consultation ID cannot be found, MAES control blocks are printed. Technical support can use these MAES control blocks to debug the problem.

13001

MAESNCON

When MAES terminates an existing consultation because a second consultation start request was received, MAES control blocks are printed. Technical support can use these MAES control blocks to further analyze the problem

13002

MAESSCIP

If MAES cannot locate the conversation ID (CID) in its internal control blocks (log message 15001), MAES logs the control blocks that can be used to analyze the problem.

14001

MAES

As part of MAES initialization, an Z/OS ESTAE function is executed to trap abends in the MAES task. If the ESTAE fails, this log message is issued and MAES terminates. Contact technical support and have both the MAES log and the Z/OS job log available.

14002

MAESDBUG

When starting the debug knowledge base, MAES issues an Z/OS IDENTIFY to make entry points within the load module known to the operating system. If the IDENTIFY fails, certain operations of the MONITOR knowledge base will not work. Contact technical support and have the MAES log and the Z/OS job log available.

14003

MAESDBUG

If the execution parameters indicate that the debug knowledge base is to be loaded, MAES attempts to bring in the debug knowledge base using the Z/OS LOAD function. If the load fails, MAES continues. Examine the Z/OS job log to determine the specific error. Usually, the problem is that neither the JOBLIB nor STEPLIB concatenations contain the MAESDBUG load module. Correct the problem before running MAES again.

14006

MAESETXR

When a consultation terminates, MAESETXR attempts to deallocate any data sets that might have been allocated by the subtask, but not deallocated by the inference engine. These data sets have a consultation-unique identifier (Annn) prefixed to the DD name. If the deallocation fails, this error message is issued with the DD name that MAESETXR failed to process.

If a de-allocation failure occurs, it can interfere with subsequent consultations because the consultation-unique identifier is reusable. The reason for the deallocation failure is found in the operating system job log for MAES. The deallocation failure is listed as a standard Z/OS error message. Examine the Z/OS error message and correct the error condition before trying to run MAES again.

14007

MAESLUNM

This log message indicates that the DD statement for MAESLUNM was missing from the JCL stream. MAES requires all potential session partners to be identified before any sessions are established. The data set associated with the MAESLUNM DD statement is scanned to identify potential session partners. Correct the error and resubmit the MAES JCL.

14008

MAESLUNM

MAESLUNM found the ACQUIRE keyword, but the argument is neither YES nor NO. The correct form is ACQUIRE(YES) or ACQUIRE(NO). Correct the MAESLUNM data set and resubmit the MAES JCL.

14009

MAESLUNM

MAESLUNM found the TYPE keyword, but the argument is not a valid selection, such as CICS or IMS. Review the MAESLUNM data set in conjunction with the associated material in this guide, and correct any errors before resubmitting the MAES JCL.

14010

MAESLUNM

Either the WIN or the MAX keyword was specified, but the TYPE keyword indicates the potential session partner is not an LU6.2 interface with parallel session support. For instance, WIN or MAX is invalid if TYPE(IMS) was specified. Correct the MAESLUNM data set and resubmit the MAES JCL.

14011

MAESLUNM

A potential session partner is being defined, but MAX was not specified for the WIN keyword. MAES needs to know the maximum number of traffic sessions that are supported so that it can maintain the appropriate number of internal control blocks. Correct the MAESLUNM data set and resubmit the MAES JCL.

14012

MAESLUNM

In processing a MAESLUNM specification for a session partner where MAES owns some of the traffic sessions, it was determined that the value from WIN exceeded the maximum number of traffic sessions as specified by MAX. Correct the problem and resubmit the MAES JCL.

14013

MAESLUNM

MAES processed the MAESLUNM data set. The MAESLUNM data set was either empty or all the entries had an associated error. In either case, correct the problems and resubmit the MAES JCL.

14014

MAESPARM

When execute parameter information is received, MAESPARM dynamically links to the MAESPC module which is used to parse the information. However, an error occurred while attempting to LINK to MAESPC. This log message is recorded and MAES terminates. Examine the Z/OS job log for the MAES job with the error messages that may be used to solve the problem. The usual cause is that the module is not in the STEPLIB or JOBLIB concatenations; however, other problems might be noted in the job log.

14015

MAESPARM

MAESPARM recognizes the SEC= execute parameter and attempts to load the indicated security program. The LOAD fails and MAES terminates. Use the Z/OS job log for the MAES job to solve the problem. In the Z/OS job log, find the Z/OS error messages that can be used to determine the problem with the security program load.

14016

MAESPRLD

MAES attempts to preload a program specified in the MAESPRLD data set. However, the program could not be loaded. The Z/OS code (such as 806) is included in this log message. The Z/OS code is consistent with the Z/OS abend codes in the *IBM Z/OS Messages and Codes* guide.

14017

MAESPRLD

An error occurred in the MAESPRLD data stream. A program name is improperly delimited and is greater than 8 bytes. The first 8 bytes of the program are included in this log message. Correct the MAESPRLD data stream and resubmit the MAES JCL.

14018

MAESSERV

A service request was made, but the internal function code was not recognized. This error represents mismatched levels between MAES and the MONITOR knowledge base, or a programming error. Verify that MAES and MONITOR are at the correct levels. If they are at the correct levels, contact technical support and submit a copy of the MAES log.

14019

MAESSHUT

If MAES intercepts an abend or program interrupt, it logs the error condition. If the debug knowledge base is available, more information is included in this log message.

14022

MAESRECP

MAES supports a fixed number of concurrent consultations (as specified in the MAXTASKS= execution parameter). This log message is issued if a start consultation request is received when the maximum number of consultations is already active.

14023

MAESINIT

MAES attempts to initialize the API, but the ADSAPI module could not be executed. Although MAES continues processing, any programs attempting to use API fail. Correct the problem by insuring that the library containing ADSAPI is included in the STEPLIB or JOBLIB concatenation.

14024

MAESINIT

The first program, INITPRGM, of the PGMSEC execution parameter could not be executed. Make sure the program is executable and is included in a STEPLIB or JOBLIB library.

14025

MAESINIT

The second program, CHECKPRGM, of the PGMSEC execution parameter could not be executed. Make sure the program is executable and is included in a STEPLIB or JOBLIB library.

14026

MAESINIT

The SAESEP version does not match the MAES version. Processing continues.

14030

MAESAPPS

This log message shows the name of the application for which pre-initialization processing was not successful. For more information, see previous log messages.

14031

MAESAPRM

An error occurred in the MAESAPRM data stream. An application name is improperly delimited and is greater than 8 bytes. The first 8 bytes of the application name are included in this log message. Correct the MAESAPRM data stream and resubmit the MAES JCL.

14032

MAESAPRM

An error occurred in the MAESAPRM data stream. An application name delimiter ("=") was not found. The first 8 bytes of the input line are included in this log message. Correct the MAESAPRM data stream and resubmit the MAES JCL.

14033

MAESAPRM

An error occurred in the MAESAPRM data stream. A null (zero length) application name was found. This is usually caused by a misplaced equal sign ("="). The application name must immediately precede the equal sign and must be between 1 and 8 bytes in length. Correct the MAESAPRM data stream and resubmit the MAES JCL.

14034

MAESAPRM

An error occurred in the MAESAPRM data stream. A terminating quote mark was not found. The parameter string must be terminated by a single quote within five card images (389 bytes is the maximum parameter length). Correct the MAESAPRM data stream and resubmit the MAES JCL.

14035

MAESAPRM

An error occurred in the MAESAPRM data stream. A space was found prior to the terminating quote mark. The parameter string must be terminated by a single quote before any spaces are encountered. Correct the MAESAPRM data stream and resubmit the MAES JCL.

14036

MAESAPRM

An error occurred in the MAESAPRM data stream. A starting quote mark was not found. The parameter string must be started by a single quote immediately after the equal sign ("=") which follows the application name. Correct the MAESAPRM data stream and resubmit the MAES JCL.

14500

SAESLOGM

Certain external programs that the inference engine calls as functions (remote KB, message switching, and so on) add log messages to the MAES log. All these log messages appear under this log message ID. Typically, they reflect errors in processing. The knowledge base receives a return code from the process object. The return code and log message in the MAES log may help you solve the problem.

14504

SAESSTIN

The checking program, CHECKPRGM, returned a nonzero value in register 15. The user is probably not authorized. The consultation abends with user abend 377.

14502

SAESRUID

SAES expects FPL data to be provided by a screen input message type with a subtype of field-by-field. SAES received a screen input message type, but it was not field-by-field data. This log message can be caused by programming errors in ADSFPL, AAESFPL, or DCFPL, or by inconsistent levels between these two programs and MAES.

14503

SAESRUID

SAES attempts to allocate enough storage to accommodate the FPL input and output data. Not enough memory is available to complete the request. If increasing the REGION parameter on the EXEC statement does not eliminate the problem, it could be the result of a programming error. In this case, contact technical support and submit the MAES log showing all the data traffic.

15000

MAESLGON

A VTAM BIND request was received, but the requester's name was not found in the internal control blocks formatted from the MAESLUNM file. If the requester is valid, the MAESLUNM data set must be upgraded to include the new session partner.

15001

MAESSCIP

This log message is produced when the VTAM SCIP exit gets control on an UNBIND request. MAES uses the VTAM conversation ID (CID) as an argument into internal control blocks. If the session was cleaned up before, it produces this log message. Typically, this is not an error situation.

15002

MAESSERV

Using a MONITOR knowledge base function, a user requested that a partner session be terminated. This log message indicates that the LU name could not be found in internal control blocks. This represents a programming error and should be reported to technical support.

15003

MAESSERV

Using a MONITOR knowledge base function, a user requested that a partner session be activated. This log message indicates that the LU name could not be found in internal control blocks. This represents a programming error and should be reported to technical support.

15004

MAESSERV

Using a MONITOR knowledge base function, a user requested that a partner session be acquired. This log message indicates that the LU name could not be found in internal control blocks. This represents a programming error and should be reported to technical support.

15005

MAESSUBS

As MAES binds traffic sessions with its session partner, the session information is stored in an internal control block. This log message is issued when more sessions need to be bound, but no more control blocks are available. This is an error condition and should be reported to technical support. Include the MAES log.

15006

MAESSERV

Using the MONITOR knowledge base, a user attempted to close and reopen the data sets associated with the MAESHPO DD statement. However, the DD statement was missing from the MAES startup JCL and the request was rejected.

15007

MAESSQ

The user attempted to reload a preloaded program using the MONITOR knowledge base and the request failed. The JES log for the MAES job contains the error information generated by Z/OS.

15008

MAESPRLD

A statement in the data set associated with the MAESPRLD DD statement is incorrect. MAES found a specification with an opening parenthesis but without a closing parenthesis. MAES assumes you have tried to use multiple statements for a single preload specification. This is not valid.

15009

MAESPRLD

While verifying the statements in the data set associated with the MAESPRLD DD statement, a priority specification was found that was not numeric. There can be no imbedded blanks, special characters, or sign information in a priority specification.

15010

MAESPRLD

A priority for a preloaded, optimized knowledge base cannot have a length greater than 2 bytes. The appropriate values are from 1 to 15, inclusive.

15011

MAESPRLD

The priority assigned to an optimized program was greater than 15.

15012

MAESPRLD

While verifying the statements in the data set associated with the MAESPRLD DD statement, a parameter specification was not recognized. The only permitted parameter values are PRIORITY=*nn* and COPIES=*nnn*.

15013

MAESPRLD

A value was not specified for the PRIORITY parameter. The appropriate values are from 1 to 15, inclusive.

15014

MAESPRLD

A value was not specified for the COPIES parameter. The appropriate values are from 1 to 999, inclusive.

15015

MAESPRLD

While verifying the statements in the data set associated with the MAESPRLD DD statement, a COPIES value was found that was not numeric. There can be no imbedded blanks, special characters, or sign information in a COPIES value.

15016

MAESPRLD

A value for the COPIES parameter cannot have a length greater than 3 bytes. The appropriate values are from 1 to 999, inclusive.

15017

MAESPRLD

A value for the PRIORITY parameter cannot be less than 1. The appropriate values are from 1 to 15, inclusive.

16000

MAESAALG

This log message indicates that an error occurred while establishing a cross-memory services session.

16001

XAESNCON

A cross-memory bind was rejected because an earlier VTAM connection had been established to the same node name. This can occur for one of the following reasons:

- MAES and the transaction driver are not at the same version level
- Another consultation is running that has locked the LU name with which you are trying to BIND
- There is a problem with the cross-memory SVC installation

16002

XAESNCON

After an XMS bind was successful, the transaction driver sent a message to MAES that was either invalid or out of sequence. Send the MAES log and any other supporting documentation to technical support.

16003

MAESSECS

See log message 16001. This is the reverse of the situation in which an XMS session was established and a transaction driver attempts to establish a VTAM session.

16004

SAESXMGE

MAES posted the consultation with a stop request. This typically happens during shutdown or in a situation in which a new consultation is to be started from the same terminal.

16005

SAESXMSE

An XMS SEND request completed, but the transaction driver indicated that there was an error. This error should be reported to technical support. Include the MAES log.

16006

XAESNCON

After an XMS bind is received, MAES attempts to establish an ENQ for the transaction driver. The ENQ failed. Send the MAES log and other supporting material to technical support.

16007

SAESXMSE

An attempt was made to send a message to the transaction driver. However, the transaction driver either failed or a protocol error occurred.

16008

XAESNCON

While trying to complete a sequence of protocol messages, the cross-memory connection to a transaction driver failed. Typically, this occurs when the transaction driver abends.

16105

SAESXMSE

A cross-memory services SEND failed.

16107

SAESXMGE

A cross-memory services GET failed.

17000

SAESAPI

An initialize API request was received from the transaction driver but the request failed. Diagnostic information is printed.

17001

SAESAPI

A request to initialize a session with the API was received but the request failed. Diagnostic information is printed.

18001

OAES

MAES has attached a new component server subtask and successfully initialized it.

18002

OAES

An error occurred in the component server processing. The component server is terminated. Also look for accompanying SAESGET/SAESSEND error messages.

18003

OAES

The client sent a cleanup request. The component server will be shut down.

18004

OAES

The component server tries to load a component DLL. Also see messages 18005 and 18006.

18005

OAES

The load of the component DLL failed. Possible reasons:

- The DLL is not in any load library accessible to MAES (via the MAESHPO concatenation).
- The DLL is accessible but invalid. Verify that the DLL was built for execution under MAES. If the DLL is built with XPLINK, ensure it has an entry in MAESAPRM specifying the XPLINK option.

18006

OAES

A component DLL was loaded successfully. The DLL handle is saved internally and will be used at shutdown time to unload the DLL.

18007

OAES

The loaded component DLL contains execution trace statements. The execution trace will be turned on. Any trace messages will go to the MAES log.

18009

OAES

The subtask name is being terminated normally.

18010

OAES

An error occurred receiving data from the client. See accompanying SAESGET error messages.

18011

OAES

An error occurred while sending data to the client. See accompanying SAESSEND error messages.

18012

OAES

An error indicator is being sent to the client because the load of the component DLL failed. This message immediately follows message 18005. Both messages include the DLL name.

18021

OAES

The component server received a function request from a client. The server queries the loaded component DLL for a function pointer using the received function name. A parameter list is built for any arguments the function expects, and then the function is called.

18022

OAES

This is for the function pointer for the requested function.

18023

OAES

This is for the data type of the functions return value, if any.

18024

OAES

This is for the number of arguments that will be passed to the function.

18025

OAES

This is for the data type for a particular argument.

18026

OAES

This message is used for memory tracking. It is issued each time the component server allocates memory for a function argument list.

18028

OAES

The component server successfully built an argument list for a function call.

18029

OAES

The component server will try to call the requested function in the component DLL.

18030

OAES

This is for the return value for a component function just called.

18031

OAES

The component server could not resolve a function pointer for the requested function. You need to verify the following conditions:

- Is the class containing the method this function represents EXPORTED?
- Is the method marked as PUBLIC?
- Is the function name spelled correctly? (Aion function names are case sensitive.)
- Was the component DLL built correctly? Also see message 18005.

18032

OAES

The component server will send the output for a function to the client with any return value and/or output arguments.

18501

VLOG

A log viewer subtask was started successfully.

18502

VLOG

A log viewer subtask will be terminated normally.

18601

RESYS10

This message contains an Aion execution trace statement.

18724

TCPIP

z/OS Open Edition was not starting. Use TSO OMVS to determine whether Open Edition can be started. PWD displays the current directory. Type **exit** to leave OMVS. If Open Edition is not operable, contact your systems programmer.

18726

TCPIP

This error occurs when another MAES region is using the same port number, or the port is unavailable. VTAM can be set up to reserve IP ports to specific jobnames. This error can occur if the MAESPARM IPPORT statement does not match the jobname specified in the VTAM profile.

Use TSO NETSTAT PORTL to see the active TCP/IP ports. See VTAM file VTAM.A00X.TCPIP(A11PROF) on CA11 to determine how TCP/IP ports are reserved to a jobname.

18736

TCPIP

The number of IPTASKS was exceeded. Check MAESPARM to ensure that enough IPTASKS are defined to meet requirements.

18743

TCPIP

No active TCP/IP address space found. Check MAESPARM to ensure that enough IPTASKS are defined to meet requirements.

19001

XAESBRPL

A node initialization block (NIB) could not be built. MAES terminates.

19002

XAESBRPL

A request parameter list (RPL) could not be built. MAES terminates.

MAES Termination Return Codes (0 - 8)

If the MAES parms are not defined properly, MAES may terminate at initialization time with a non-zero return code indicating the nature of the failure:

0

Return Code

Indicates successful termination.

7

Return Code

Indicates that MAES can not open its ACB using the NODENAME value specified in the MAES parms.

8

Return Code

Indicates that one of the MAES parms (other than NODENAME) is invalid. Check the ones that changed since the last time MAES was executed.

CICS and IMS Client Return Codes (00 - 99)

All CICS and IMS client calls to MAES include a return code and reason code, which the client program can inspect to determine the outcome of the requested operation. A return code of zero indicates success while a non-zero return code indicates the nature of the failure. Where applicable, the reason code contains the error code associated with the failure. In most cases, examination of the return code alone is sufficient to diagnose the problem. However, in other cases (most notably return code 99), analysis of the reason code is required. The reason codes generated by MAES are included in the next section. For other operations (such as Allocate), see the associated IBM documentation to determine the meaning of the associated reason code.

00

Return Code

OK

04

Return Code

A CICS or IMS client program using the Suspend communication protocol attempted one of the function calls (for example, SUSPEND, RESUME, ACQUIRE, or RELEASE) which are only supported by the non-Suspend communication protocol.

- For COBOL and PL/1 clients, relink the client program using the non-Suspend SDS (see the comment block at the start of the sample client programs with names ending in "2" or "3" for details).
- For C clients, specify the CONVERSATIONAL parm in the remote or batch build options so that the generated client DLL uses the non-Suspend communication protocol.

08

Allocate failed

Connect failed. The client could not allocate an LU6.2 session to MAES.

16

SEND Resume request failed

A problem occurred while trying to send a RESUME request to an active component.

20

RECEIVE Resume response failed

A RESUME request has been sent, but the expected RESUME OK message was not received.

24

SEND Suspend request failed

A problem occurred while trying to send a SUSPEND request to an active component.

28

RECEIVE Suspend response failed

A SUSPEND request has been sent, but the expected SUSPEND OK message was not received.

32

SEND (generic) failed

A problem occurred while trying to send a request to MAES.

36

RECEIVE (generic) failed

A problem occurred while receiving from MAES.

40

SEND OAES Start command failed

A problem occurred while sending a startup command for OAES to MAES.

44

RECEIVE Start OK failed

An OAES startup command has been sent, but the START OK message was not received.

48

RECEIVE OAES Echo failed

An OAES component server has been started, but it could not send its startup confirmation.

52

SEND Load DLL failed

A problem occurred while trying to send a LOAD DLL request to OAES.

56

RECEIVE Load DLL failed

A LOAD DLL request has been sent, but the corresponding DLL handle was not received.

60

NULL Instance received

A _create function request has been sent, but no instance handle was received.

64

NULL Function pointer received

A function call request has been sent, but OAES could not resolve the function pointer.

70

Memory allocation error

The client wrapper could not obtain memory for a request.

80

MAES node name error

The first character of the MAES node name is blank or null.

90

Copybook processing error, Reason code 0 or 1

- Reason code 0 indicates an unrecognized parameter type. This error is usually caused by an out-of-date or manually altered copybook.
- Reason code 1 indicates an excessive message length (approximately 32K for CICS or 64K for IMS). This error may be caused by an out-of-date or manually altered copybook, but it also could reflect an attempt to pass too much data in one method call.

99

MAES Error, Reason code provided

An error was received from MAES. See the corresponding reason code.

CICS and IMS Client Reason Codes

CICS and IMS Client Reason Codes consist of the following:

- MAES Reason Codes
- Data Access Reason Codes
- IMS Reason Codes
- CICS Reason Codes

MAES Reason Codes

Internal Reason Codes	Description
#LODFAIL EQU 210	REQUEST TO LOAD A DLL FAILED
#ABEND EQU 302	ABNORMAL END-OF-SESSION
#TIMEOUT EQU 303	TASKETTE TIMEOUT
#NOCON EQU 309	NO CONSULTATION IN PROGRESS
#NOTASKS EQU 310	NO MORE TASKETTES AVAILABLE
#BADMREQ EQU 311	UNEXPECTED SUB-MESSAGE TYPE
#BADSESS EQU 312	INVALID SESSION PARMS ON BIND
#NOSTORE EQU 313	OUT OF STORAGE STARTING CONSULT
#BADVTAM EQU 314	UNEXPECTED VTAM ERROR
#BADCREQ EQU 315	BAD CONVERSATION REQUEST
#SHUTDWN EQU 316	MAES IN SHUTDOWN - REQUEST REJECTED
#TASKMEM EQU 317	TOO MUCH MEMORY REQUESTED BY SUBTASK
#BADLUNM EQU 318	LU NAME NOT IN MAES' TABLE
#CONVXOK EQU 324	APPLICATION TASK ENDED (NO ABEND CODE)
#BISSENT EQU 326	'BRACKET INIT STOP' SEND UNEXPECTEDLY
#UNKDR EQU 327	UNKNOWN DATA REQUEST TYPE
#SSA2MNY EQU 329	> 16 SSA'S IN A DL/I REQUEST
#BADDBN EQU 330	UNKNOWN DL/I DATABASE NAME
#NODATA EQU 331	DATA ELEMENT MISSING
#BADLEN EQU 332	BAD ITEM LENGTH IN MESSAGE DATA
#BAD2CUR EQU 352	BAD CURSOR NAME
#BAD2CNM EQU 353	BAD CURSOR NUMBER
#BADMSGT EQU 354	UNEXPECTED MESSAGE TYPE
#BADFUNC EQU 355	BAD IMBEDDABILITY FUNCTION TYPE
#BADPNUM EQU 357	INVALID # OF PARMS FOR IMBED IFACE
#IMOPEN EQU 358	TRYING TO OPN SESS W/O CLOSE (IMI)
#NOOPSPC EQU 360	NO SPACE FOR OUTPUT PARMS

Internal Reason Codes	Description
#IDTOBIG EQU 361	A SUPPLIED IDENTIFIER IS TOO BIG
#NODIACT EQU 365	NODE INACTIVE FOR A BIND REQ.
#BADSERV EQU 366	BAD SERVICE REQUEST CODE
#PROTFMH EQU 386	TRIED TO SEND FMH5 IN MIDDLE OF RU
#LINEOUT EQU 388	SESSION WAS LOST (LINE IS OUT)
#PROTBRK EQU 389	SEND ATTEMPTED BETWEEN BRACKETS
#NOHPO EQU 390	NO MAESHPO DD STATEMENTOK
#DB2COLS EQU 398	DB2 COLS > THAN REQUEST
#DUPECON EQU 399	DUPLICATE CONSULTATION
#B62NAME EQU 900	BAD NODE NAME
#B62TYPE EQU 902	WRONG TYPE ... NOT LU6.2 NODE
#B62SLOT EQU 903	NO SLOTS AVAILABLE
#B62INAC EQU 904	THE NODE ISN'T ACTIVE
#B62NSAL EQU 905	NO SUBSESSIONS BOUND
#B62NCID EQU 906	CIDD IS NO GOOD
#B62NOWN EQU 907	SUBSESSION IS NOT ALLOCATED
#B62NMIN EQU 908	SUBSESSION IS NOT ALLOCATED TO ME
#B62CL0 EQU 909	COMMAND LENGTH IS ZERO
#B62CTB EQU 910	1ST WORD OF COMMAND > 8 BYTES
#B62CLB EQU 911	LEADING BLANKS ON COMMAND
#B62BBLD EQU 912	'$ADS' IN THE COMMAND
#B62CTL EQU 913	COMMAND IS TOO LONG FOR BUFFER
#B62SEND EQU 914	'SEND' WHEN IN 'RECEIVE' MODE
#B62FPL5 EQU 915	FPL DATA REQUESTED AFTER FMH5 SENT
#B62CREC EQU 916	COMMUNICATIONS BLOCK IS MISSING
#B62BST EQU 917	BAD CONSULTATION START
#SSPGM EQU 1350	UNABLE TO LOAD THE SSQL PROGRAM

Data Access Reason Codes

VSAM Reason Codes	Description
#VSMBREQ EQU 1206	INVALID VSAM FILE REQUEST
#IP2BIGL EQU 1207	INVALID INPUT PARAMETER LENGTH
#IP2BIG# EQU 1208	TOO MANY INPUT PARAMETERS
#VSMNOID EQU 1210	NO VSAM RECORD ID
#VSMNOKY EQU 1219	NO KEY SPECIFIED (A ZERO LENGTH KEY)
#VSMBFLG EQU 1220	INVALID INPUT DATA FLAG
#VSMBDAT EQU 1221	INVALID INPUT DATA
#VSMBKEY EQU 1222	INPUT KEYS CONFLICT

DL/1 Reason Codes	Description
#DLDBOPE EQU 1300	DL/1 NOT OPEN
#DLREQE EQU 1301	DL/1 INVALID REQUEST
#DLPCBE EQU 1302	DL/1 INVALID PCB ADDRESS
#DLSCHED EQU 1304	DL/1 SCHEDULING CONFLICT
#DLNOPSB EQU 1306	DL/1 PSB NOT FOUND
#DLTASK EQU 1307	DL/1 TASK NOT AUTHORIZED
#DLACTIV EQU 1308	DL/1 PSB ALREADY SCHEDULED
#DLLANG EQU 1309	DL/1 LANGUAGE CONFLICT
#DLPSBE EQU 1310	DL/1 PSB INITIALIZATION FAILED
#DLAUTH EQU 1311	DL/1 PSB NOT AUTHORIZED
#DLTERM EQU 1312	DL/1 TERMINAL UNSCHEDULED
#DLFUNC EQU 1313	DL/1 FUNCTION UNSCHEDULED
#DLDOWN EQU 1314	DL/1 NOT ACTIVE
#DLFCTR EQU 1315	DL/1 UIBFCTR UNKNOWN
#DLDLTR EQU 1316	DL/1 UIBDLTR UNKNOWN
#DLDATA EQU 1317	DL/1 DATA FOUND
#DLNDATA EQU 1318	DL/1 DATA NOT FOUND

IMS Reason Codes

Common Reason Codes	Description
@BIGREC EQU 12	RECORD TOO BIG FROM 'SAESGET'
@SMALLAR EQU 16	GETMAIN FOR I/O AREA FAILED (IAESALEN)
@BADSESS EQU 20	VTAM SESSION INACTIVE - REQUEST IGNORED
@NEGLEN EQU 24	NEGATIVE DATA LENTH SPECIFIED ON SEND
@DATAREQ EQU 28	DATA REQUEST ERROR OF SOME SORT
@BLANKLU EQU 32	LUNAME WAS ALL BLANKS (IAESSTIN)
@BADOPEN EQU 36	ACB DIDN'T OPEN IAESSTIN)
@BADDST EQU 40	BAD 'OPNDST' (IAESSTIN)
@NOTOPEN EQU 44	ACB WASN'T OPEN (IAESTERM)
@BADCLS EQU 48	BAD 'CLSDST' (IAESTERM)
@BADSEND EQU 52	BAD 'SEND' (IAESTERM)
@BADREC EQU 56	BAD 'RECEIVE' (IAESTERM)
@INACT EQU 60	INCOMPLETE ACTION IN SUBROUTINE
@STOPCON EQU 64	CONSULTATION STOPPED - INTERNAL ERROR
@UNKMSG EQU 68	UNKNOWN MESSAGE TYPE RECEIVED
@VTAMERR EQU 72	VTAM PROBLEM OCCURRED
@NOLUNME EQU 76	NO LUNAME SPECIFIED FOR FPL
@MAESDED EQU 84	SELECTED MAES IS INACTIVE
@NOLUNM EQU 88	LUNAME IS ALREADY IN USE

CICS Reason Codes

CICS is maintained separately from Aion and its reason codes may change. See IBM documentation on the function indicated by the client return code for a complete and current explanation of the potential reason code values.

Common Reason Codes	Description
00	NORMAL
01	ERROR
02	RDATT

Common Reason Codes	Description
03	WRBRK
04	EOF
05	EODS
06	EOC
07	INBFMH
08	ENDINPT
09	NONVAL
10	NOSTART
11	TERMIDERR
12	FILENOTFOUND
13	NOTFND
14	DUPREC
15	DUPKEY
16	INVREQ
17	IOERR
18	NOSPACE
19	NOTOPEN
20	ENDFILE
21	ILLOGIC
22	LENGERR
23	QZERO
24	SIGNAL
25	QBUSY
26	ITEMERR
27	PGMIDERR
28	TRANSIDERR
29	ENDDATA
30	INVTSREQ
31	EXPIRED
32	RETPAGE

Common Reason Codes	Description
33	RTEFAIL
34	RTESOME
35	TSIOERR
36	MAPFAIL
37	INVERRTERM
38	INVMPSZ
39	IGREQID
40	OVERFLOW
41	INVLDC
42	NOSTG
43	JIDERR
44	QIDERR
45	NOJBUFSP
46	DSSTAT
47	SELNERR
48	FUNCERR
49	UNEXPIN
50	NOPASSBKRD
51	NOPASSBKWR
52	(Not Defined)
53	SYSIDERR
54	ISCINVREQ
55	ENQBUSY
56	ENVDEFERR
57	IGREQCD
58	SESSIONERR
59	SYSBUSY
60	SESSBUSY
61	NOTALLOC
62	CBIDERR

Common Reason Codes	Description
63	INVEXITREQ
64	INVPARTNSET
65	INVPARTN
66	PARTNFAIL
67	(Not Defined)
68	(Not Defined)
69	USERIDERR
70	NOTAUTH
71	VOLIDERR
72	SUPPRESSED
73	(Not Defined)
74	(Not Defined)
75	RESIDERR
76	(Not Defined)
77	(Not Defined)
78	(Not Defined)
79	(Not Defined)
80	NOSPOOL
81	TERMERR
82	ROLLEDBACK
83	END
84	DISABLED
85	ALLOCERR
86	STRELERR
87	OPENERR
88	SPOLBUSY
89	SPOLERR
90	NODEIDERR
91	TASKIDERR
92	TCIDERR

Common Reason Codes	Description
93	DSNNOTFOUND
94	LOADING
95	MODELIDERR
96	OUTDESCRERR
97	PARTNERIDERR
98	PROFILEIDERR
99	NETNAMEIDERR
100	LOCKED
101	RECORDBUSY
102	UOWNOTFOUND
103	UOWLNOTFOUND
104	LINKABEND
105	CHANGED
106	PROCESSBUSY
107	ACTIVITYBUSY
108	PROCESSERR
109	ACTIVITYERR
110	CONTAINERERR
111	EVENTERR
112	TOKENERR
113	NOTFINISHED
114	POOLERR
115	TIMERERR
116	SYMBOLERR
117	TEMPLATERR
118	(Not Defined)

CA Health Checker Messages

Provides a simple and consistent method for CA products to create health checks to run under the IBM Health Checker for z/OS. The IBM Health Checker for z/OS helps you identify potential problems in your z/OS environment by checking system or product parameters and system status against recommended settings. CA has joined other vendors in creating checks for CA z/OS products. CA Aion BRE health checks are automatically activated on the target system when the product is started on a system where the following components are installed and configured:

- CA Health Checker Common Service
- IBM Health Checker for z/OS

For more information on installing the CA Health Checker Common Service, see the CA Common Service Installation Guide.

For more information about the IBM Health Checker for z/OS, see the IBM Health Checker for z/OS User Guide.

AIONBRE01W

<mv class=decimal>busypct</mv>% of VTAM requests were rejected because all tasks were busy.

Reason:

There are <mv class=decimal>maxtasks</mv> VTAM server tasks active. <mv class=decimal>nrequests</mv> VTAM requests were received during the last check interval. There were <mv class=decimal>nbusy</mv> busy responses.

This message appeared because the busy exception threshold (BUSYPCTLVL) was exceeded. The BUSYPCTLVL parameter is set to: <mv class=decimal>busypctlvl</mv>%.

MAES services VTAM requests with <mv class=decimal>maxtasks</mv> tasks. When all of these tasks are busy performing other requests, new requests are rejected. This problem requires users to send requests again. If requests are sent programmatically there is a potential for communication loss. If the number of busy responses becomes excessive MAES may need to be restarted. The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive.

Action:

Ensure adequate MAES tasks are active to service arriving VTAM requests.

System Action:

MAES rejects VTAM requests with busy responses.

Operator Action:

If your installation permits, you can alter the number of VTAM service tasks by the following command:

```
F <mv>maesjobname</mv>,MAXTASKS=nn
```

Alternatively, you can alter the busy threshold (<mv class=decimal>busypctlvl</mv>%) by the following command:

```
F <mv>maesjobname</mv>,BUSYPCTLVL=nn
```

The MAXTASKS and BUSYPCTLVL parameters can also be modified in the MAES startup parameter file.

Refer to the MAESLOG file of job <mv>maesjobname</mv> to determine the MAESPARM file that is referenced during MAES startup.

Refer to the SYSOUT file of job <mv>maesjobname</mv> to determine the parameter file member (PMEMBER) that is used during MAES startup.

System Programmer Action:

Study the value of the MAES MAXTASKS parameter. Determine if a higher number of tasks can be used. Some MAES applications require a considerable amount of memory, which might require keeping the maximum number of tasks at a specific threshold.

Problem Determination Action:

Use the MAES Monitor to study the active tasks, and the number of requests that are arriving.

Source:

CA Aion Business Rules Expert (BRE) Multitasking Aion Execution System (MAES) -- MAESHC01

Reference Documentation:

Refer to the description of the MAXTASKS parameter in the Aion BRE Mainframe User's Guide.

AIONBRE02W

<mv class=decimal>busypct</mv>% of VTAM requests were rejected because all tasks were busy.

Reason:

There are <mv class=decimal>maxtasks</mv> VTAM server tasks active. This message is a warning that some busy responses are occurring. <mv class=decimal>nrequests</mv> VTAM requests were received during the last check interval. There were <mv class=decimal>nbusy</mv>busy responses.

The busy exception threshold (BUSYPCTLVL) is set to <mv class=decimal>busypctlvl</mv>%.

An exception health check message is not presented until the busy exception threshold is exceeded.

The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive.

Action:

Ensure adequate MAES tasks are active to service arriving VTAM requests.

AIONBRE03I

MAES is able to process all arriving VTAM requests without sending any busy responses.

Reason:

There are <mv class=decimal>maxtasks</mv> VTAM server tasks active. <mv class=decimal>nrequests</mv> VTAM requests were received during the last check interval. The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive. MAES is operating normally.

Action:

Ensure adequate MAES tasks are active to service arriving VTAM requests.

AIONBRE05W

<mv class=decimal>busypct</mv>% of TCP/IP requests were rejected because all tasks were busy.

Reason:

There are <mv class=decimal>iptasks</mv> TCP/IP server tasks active. <mv class=decimal>nrequests</mv> TCP/IP requests were received during the last check interval. There were <mv class=decimal>nbusy</mv> busy responses. This message appeared because the busy exception threshold (BUSYPCTLVL) was exceeded. The BUSYPCTLVL parameter is set to: <mv class=decimal>busypctlvl</mv>%.

MAES services TCP/IP requests with <mv class=decimal>iptasks</mv> tasks. When all of these tasks are busy performing other requests, new requests are rejected. This problem requires users to send requests again. If requests are sent programmatically there is a potential for communication loss. If the number of busy responses becomes excessive MAES may need to be restarted. The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive.

Action:

Ensure adequate MAES tasks are active to service arriving TCP/IP requests.

System Action:

MAES rejects TCP/IP requests with busy responses.

Operator Action:

If your installation permits, you can alter the number of TCP/IP service tasks by the following command:

```
F <mv>maesjobname</mv>,IPTASKS=nn
```

Alternatively, you can alter the busy threshold (<mv class=decimal>busypctlvl</mv>%) by the following command:

```
F <mv>maesjobname</mv>,BUSYPCTLVL=nn
```

The IPTASKS and BUSYPCTLVL parameters can also be modified in the MAES startup parameter file. Refer to the MAESLOG file of job <mv>maesjobname</mv> to determine the MAESPARM file that is referenced during MAES startup.

Refer to the SYSOUT file of job <mv>maesjobname</mv> to determine the parameter file member (PMEMBER) that is used during MAES startup.

System Programmer Action:

Study the value of the MAES IPTASKS parameter. Determine if a higher number of tasks can be used. Some MAES applications require a considerable amount of memory, which might require keeping the maximum number of tasks at a specific threshold.

Problem Determination:

Study MAES sysout to review TCP/IP activity.

Source:

CA Aion Business Rules Expert (BRE) Multitasking Aion Execution System (MAES) -- MAESHC01

Reference Documentation:

Refer to the description of the IPTASKS parameter in the Aion BRE Mainframe User's Guide.

AIONBRE06W

<mv class=decimal>busypct</mv>% of TCP/IP requests were rejected because all tasks were busy.

Reason:

There are <mv class=decimal>iptasks</mv> TCP/IP server tasks active. <mv class=decimal>nrequests</mv> TCP/IP requests were received during the last check interval. There were <mv class=decimal>nbusy</mv> busy responses. This message is a warning that some busy responses are occurring. The busy exception threshold is set to <mv class=decimal>busypctlvl</mv>%.

An exception health check message is not presented until the busy exception threshold is exceeded. The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive.

Action:

Ensure adequate MAES tasks are active to service arriving TCP/IP requests.

AIONBRE07I

MAES is able to process all arriving TCP/IP requests without sending any busy responses.

Reason:

There are <mv class=decimal>iptasks</mv> TCP/IP server tasks active. <mv class=decimal>nrequests</mv> TCP/IP requests were received during the last check interval. The check interval is 5 minutes normally, and 1 minute while the busy response percent level is excessive. MAES is operating normally.

Action:

Ensure adequate MAES tasks are active to service arriving TCP/IP requests.

Index

A

B

C

D

E

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z