User Guide: How to use the WAS Bridge or make OLA calls in ECB-controlled programs

OPTIMIZED LOCAL ADAPTERS (OLA) AND ALCS

WAS optimized local adapters are built-in, high speed, bidirectional connectors that allow a z/OS application, such as ALCS, to communicate with WebSphere Application Server for z/OS J2EE applications. That is, they allow calling between z/OS WebSphere and ALCS on the same z/OS image. They provide a strong differentiator for hosting WebSphere on z/OS alongside existing ALCS applications. With this support, customers are provided a new high performing mechanism to support efficient integration of newer Java-based applications with ALCS applications. Performance will be significantly better than any equivalent function. Also this support eliminates the need for ALCS customers to have to write their own connector code and to parse their messages to and from XML/SOAP format in the ALCS environment.

With ALCS WebSphere OLA support:

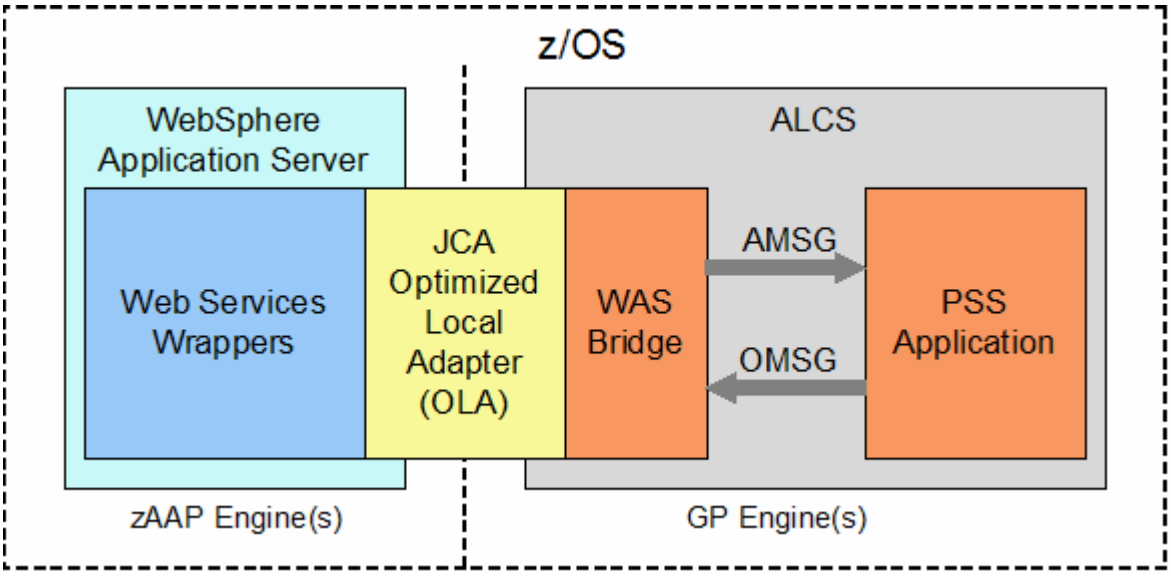
- Extremely efficient communication is provided by using shared memory services
- WebSphere Application Server can utilize cost efficient zAAP engines
- Co-location of WebSphere Application Server and ALCS provides superior reliability
- OLA uses standard J2EE programming paradigms

ALCS WAS BRIDGE AND OPTIMIZED LOCAL ADAPTERS

The ALCS WAS Bridge provides connectivity between current ALCS applications and WebSphere applications. It allows input messages to be formatted and passed on to legacy applications as if they came from ordinary devices. Output ALCS application messages can also be routed to the WAS Bridge.

Existing ALCS applications can continue to use the traditional ALCS APIs by taking advantage of the ALCS WAS Bridge.

See below for more details:



ALCS ECB-CONTROLLED PROGRAMS AND OPTIMIZED LOCAL ADAPTERS

You can include OLA callable services in assembler and C/C++ application programs. When calling to WAS, the target in WAS is an Enterprise Java Bean (EJB), which could make client requests in a stateful way to other WAS applications. WAS calling ALCS applications is also supported.

Here is an example of an OLA call in an assembler ECB-controlled application program:

```
CALL  BBOA1REG,(Daemon_Name,
                Node_Name,
                Server_Name,
                Register_Name,
                Minconn,
                Maxconn,
                Sec,
                Rc,
                Rsn),VL,MF=(E,APIPlist)
```

GENERATING THE SYSTEM CONFIGURATION TABLE

There is a new, optional WAS parameter on the ALCS system generation SCTGEN macro. When you require OLA then you must change your ALCS generation and you must **restart** your ALCS system.

WAS={NO|YES|(YES,1|*threads*)}

Where:

NO
Communication resources used by the WAS bridge **cannot** be started.
Application programs **cannot** issue WAS OLA callable services.
(This is the default).

YES|(YES,1|*threads*)
Communication resources used by the WAS bridge **can** be started.
Application programs **can** issue WAS OLA callable services.
ALCS allows up to *threads* concurrent OLA calls from application programs.
The default for *threads* is 1; the maximum is 2000.

Specify **WAS=YES** if you use WAS Bridge communication resources but do not use OLA callable services.

Note that you have to put the library which contains the OLA (stub) load modules in an accessible link list library before you restart ALCS. If the LNKAUTH parameter on your IEASYS*xx* parmlib member does not specify (or default to) LNKAUTH=LNKLST, you have to APF authorize that OLA load module library. Failing to do so will result in abend U0042 during restart (the ALCS initializer identified an error that prevents successful initialization).

GENERATING A COMMUNICATION LOAD MODULE WHEN USING THE WAS BRIDGE

There are two new COMDEF LDTYPE values for the WAS Bridge communication resources (LDTYPE=WAS and LDTYPE=WASTERM).

Here is an example which defines a WAS Bridge connection with two terminals:

```
*
*         WAS BRIDGE COMMUNICATION RESOURCES
*
          COMDFLT CLEAR
          COMDFLT LDTYPE=WAS,UPDATE=ADD
*
           COMDEF NAME=BRIDGEW1,ISTATUS=ACTIVE,
                  APPL=RES0,
                  WASNAME=(SY1,SY1,BBOS001),
                  REGNAME=ALCSWASPIPE1,
                  SERVNAME=(alcsz001,
                  ejb/com/ibm/alcs/alcsoutbound)
*
*
*         WAS BRIDGE TERMINALS
*
          COMDFLT CLEAR
          COMDFLT LDTYPE=WASTERM,UPDATE=ADD
*
          COMDEF NAME=BRIDGET1,LINK=BRIDGEW1,
                APPL=RES0,TERM=4505
*
          COMDEF NAME=BRIDGET2,LINK=BRIDGEW1,
                APPL=RES0,TERM=3270DSP
```

Note especially the *registername* and *requestservicenames* restrictions imposed by ALCS.

REGNAME=*registername*

The name of the set of OLA local connections for this WAS resource, 1 to 12 upper-case alphanumeric characters. This name must be unique within the GRS complex.

SERVNAME=(*requestservicename_1*,*requestservicename_2***)**

requestservicenname_1

The name of the target service specified on the InteractionSpec by the WAS application, 1 to 40 characters. This must be unique within this ALCS system.

requestservicenname_2

The name of the WAS service (JNDI home name for the target EJB), 1 to 40 characters. This must be unique in the WAS server you are targeting.

The service names can contain mixed-case alphanumeric characters and any other characters that are allowed for WAS object names (but not imbedded spaces).

You can combine upper-case and lower-case characters freely in the service names. For example:

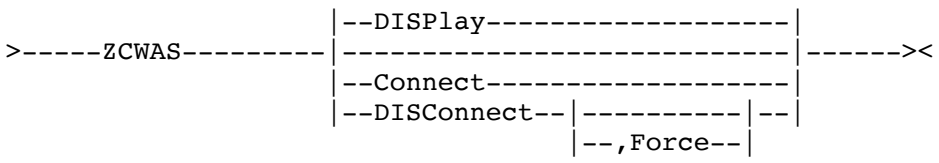
```
SERVNAME=(ALCSZ001,
ejb/com/ibm/alcs/alcsoutbound)
```

ZCWAS COMMAND

There is a new ZCWAS command to establish or terminate the connection between ALCS and WAS, or display the status of the connection.

Use the ZCWAS command (from a Prime CRAS authorized terminal only) to establish or terminate the connection between ALCS and WAS. Use the ZCWAS command (from any CRAS authorized terminal) to display the connection status.

The format of the command is:



Where:

Connect
Connect ALCS and WebSphere Application Server for z/OS interface.

DISConnect
Disconnect ALCS after inactivating all active WAS bridge communication resources. If threads are active in WAS OLA callable services, a disconnect command without the Force option does not complete.

Force
Forcibly disconnect ALCS even though threads are still active in WAS OLA callable services. The Force option may be used only after a disconnect command without the Force option has failed to complete.

DISPlay
Display status information. This is the default for ZCWAS with no additional parameters.

ALCS INSTALLATION-WIDE MONITOR EXITS

WAS authorization exit — USRWAS1

ALCS calls this exit before it processes any OLA callable services (of the form BBOA1xxx) made by ALCS ECB-controlled applications.

Use this exit to validate or restrict (or both) authorization for the OLA call, according the originating terminal address.

You may also use this exit to identify an "input" or "output" message. This causes the count of input or output WAS messages to be incremented. ZSTAT MSG or ZSTAT ALL displays these counts. If data collection is currently collecting statistics about input and output messages, this also causes the statistics for input or output WAS messages to be incremented.

WAS input bridge address exit — USRWAS3

Use this exit to identify the originating terminal. By default, the WAS input bridge assumes there is no originating terminal; it uses the WAS resource as the originator instead. Use this exit when another technique is used to identify the originating terminal. For example a CRI is contained in an 8-byte correlator preceding the message. ALCS will remove the 8-byte correlator on return from this exit.

WAS input bridge format exit — USRWAS4

By default, the WAS input bridge assumes that the message does not require any reformatting. Use this exit when reformatting is required. USRWAS4 must be sensitive to the format of the message.

WAS output bridge address exit — USRWAS5

Use this exit to add an 8-byte correlator which identifies the destination terminal. ALCS will add the 8-byte correlator in front of the message on return from this exit.

WAS output bridge format exit — USRWAS6

By default, the WAS output bridge assumes that the message does not require any reformatting. Use this exit when reformatting is required. USRWAS6 must be sensitive to the format of the message.

DETAILED INFORMATION ABOUT ALCS ECB-CONTROLLED APPLICATION PROGRAMS

ALCS ECB-controlled applications may invoke OLA callable services (of the form BBOA1xxx). When calling to WAS, the target in WAS is a stateless session EJB, which could make client requests in a stateful way to other WAS applications. When calling from WAS, the WAS application could be stateful or stateless; it could also be a servlet and not even an EJB. This means, for example, that ALCS customers can create a web service to represent some business logic that is implemented as a servlet or an EJB, and the application that gets control for the web service call can simply delegate to a connector call to ALCS. Using this approach, ALCS applications can readily be exposed externally as web services since all the necessary WAS external support and administration is in place.

Be aware that the WAS programming model is quite different from current ALCS practice. For example you can "register" via the local WAS to any WAS anywhere, and multiple applications can be involved. The so-called OLA register names and connection handles must be maintained by the applications themselves. ALCS will not unregister, or release a connection, when an ECB terminates.

For detailed information about including OLA callable services (of the form BBOA1xxx) in application programs, and for information about the characteristics of these programs, see
http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.zseries.doc/info/welcome_nd.html
Note that ALCS does not support the use of the OLA ASYNC, TRANS, and SEC parameters.

Here is an example of an OLA call in an ALCS assembler program:

```
CALL  BBOA1REG,(Daemon_Name,
      Node_Name,
      Server_Name,
      Register_Name,
      Minconn,
      Maxconn,
      Registerflags,
      Rc,
      Rsn),VL,MF=(E,EBX000)
```

Here is an example of an OLA call in an ALCS C/C++ program:

```
int bboa1reg ();
int alcs_rc;
...
...
...
alcs_rc = BBOA1REG ( &dgn,
                     &nodename,
                     &servername,
                     &regname,
                     &minconn,
                     &maxconn,
                     &regflags,
                     &rc,
                     &rsn
                   );
```

Note that you must test the ALCS return code (alcs_rc). That return code is provided to an assembler program in general register 15 and to a C/C++ program as the integer return value from the function call.

ALCS return codes are:

00 OK (Call is executed. *You still must check the OLA rc and rsn*.)
08 WAS is not enabled
12 WAS is not connected
16 Caller is not authorized
20 Not executed as WAS is forcibly disconnected

Also note that you must link-edit those ALCS ECB-controlled programs with the ALCS ECB-controlled program CDSN.

The BBOA1INV and BBOA1SRV calls are composite calls, the other BBOA1xxx calls are more primitive, more flexible calls.

Note that in the examples below the register, unregister, and get and release connection handle calls are not typically shown.

a) Invoke**:** send request message and receive response message

BBOA1INV

---> a

b) Host a service**: r**eceive request message, send response message, **and** release the connection handle obtained by BBOA1SRV

BBOA1SRV
BBOA1SRP
BBOA1CNR

---> b

c) Receive request (specific) message and send response message

BBOA1RCS
BBOA1GET
BBOA1SRP

---> c

d) Receive request (any) message and send response message

BBOA1RCA
BBOA1GET
BBOA1SRP

---> d

e) Send request message and receive response

BBOA1SRQ
BBOA1GET

---> e

AN EXAMPLE OF AN ECB-CONTROLLED PROGRAM

```
*--------------------------------------------------------------------*
*        ALCS - WAS/OLA EXAMPLE                                      *
*--------------------------------------------------------------------*
*        TEST BBOA1REG, BBOA1INV, AND BBOA1URG                       *
*                                                                    *
*        NB. THIS IS ONLY AN EXAMPLE! THEREFORE THIS PROGRAM         *
*        CALLS BBOA1REG, BBOA1INV, AND BBOA1URG IN **ONE** STEP      *
*                                                                    *
*                                                                    *
*        DGN  =  SY1                                                 *
*        NODE =  SY1                                                 *
*        SRVR =  BBOS001                                             *
*        REGNM = HSCBA012                                            *
*        OUTBOUND SERVNME = ejb/com/ibm/ola/olasample1_echoHome      *
*                                                                    *
*        TEST MUST BE LINK EDITED TOGETHER WITH CDSN                 *
*                                                                    *
*        Input ZDRIV TEST                                            *
*                                                                    *
*        Required: An EJB which sends back a "reponse"               *
*--------------------------------------------------------------------*
         EJECT ,
         BEGIN NAME=TEST,VERSION=00,TYPE=IPARS,AMODE=31,              -
               SHR=NONE,XCL=NONE
         SPACE 1
******** OBTAIN HEAP STORAGE
         SPACE 1
         L     R02,=A(10000)           SIZE OF HEAP STORAGE
         SPACE 1
         MALOC SIZE=R02                REQUEST HEAP STORAGE
         SPACE 1
         USING MALOC_AREA,R02          R02 POINTS TO STORAGE AREA
         EJECT ,
******** SET UP PARAMETERS FOR REGISTER
         SPACE 1
         MVC   Daemon_Name(8),DGN      COPY DGN
         MVC   Node_Name(8),NODE       COPY NODE NAME
         MVC   Server_Name(8),SRVR     COPY SERVER NAME
         MVC   Register_Name(16),RGN   COPY REGISTER NAME
         LA    R3,1                    NUMBER OF MIN. CONNECTIONS
         LA    R5,10                   NUMBER OF MAX. CONNECTIONS
         ST    R3,Minconn              SET NUMBER OF MIN. CONNECTIONS
         ST    R5,Maxconn              SET NUMBER OF MAX. CONNECTIONS
         SR    R15,R15                 SET TO ZEROES
         ST    R15,Sec                 SET Sec TO ZEROES
         ST    R15,Rc                  SET RC TO ZEROES
```

```
         ST    R15,Rsn                 SET Rsn to ZEROES
         SPACE 1
******** CALL BBOA1REG AND REGISTER WITH WAS OLA
         SPACE 1
         CALL  BBOA1REG,(Daemon_Name,                                  *
               Node_Name,                                              *
               Server_Name,                                            *
               Register_Name,                                          *
               Minconn,                                                *
               Maxconn,                                                *
               Sec,                                                    *
               Rc,                                                     *
               Rsn),VL,MF=(E,APIPlist)
         SPACE 1
         LTR   R15,R15                 ALCS RETUN CODE IS ZEROES
         BNZ   BADBAD                  BRANCH IF NOT - ERROR
         SPACE 1
         L     R0,Rsn                  LOAD REASON CODE
         L     R15,Rc                  LOAD RETURN CODE
         LTR   R15,R15                 REGISTER OK
         BZ    TEST010                 BRANCH IF YES
         SPACE 1
ERR1     SERRC E,BAD001                REGISTER FAILED                 *
                                       CHECK R15 AND R00
         EJECT ,
******** SET UP PARAMETERS FOR INVOKE AND TRANSLATE MESSAGE TO ASCI
         SPACE 1
TEST010  DC    0H'0'
         LA    R15,1                   LOAD "REQUEST TYPE 1'
         ST    R15,Request_Type        SET REQUEST TYPE
         MVI   Service_Name,X'40'      SET TO BLANK
         MVC   Service_Name+1(254),Service_Name SET REMAINDER TO BLANK
         MVC   Service_Name+0(35),SRVNAME SET SERVICE NAME
         LA    R15,35                  LOAD LENGTH OF SERVICE NAME
         ST    R15,Service_Namelen     SET LENGTH OF SERVICE NAME
         MVI   Request_Area,C' '       SET TO BLANK
         MVC   Request_Area+1(254),Request_Area SET REMAINDER TO BLANK
         MVC   Request_Area+0(26),=CL26'Hey WAS from nonLE assmblr'
         LA    R14,256                 LENGTH OF AREA
         SPACE 1
         ASCIC TO,DATA=Request_Area,LENGTH=(R14) TRANSLATE TO ASCI
         SPACE 1
         MVI   Response_Area,C' '      SET TO BLANK
         MVC   Response_Area+1(254),Response_Area SET REMAINDER
         LA    R15,256                 LENGTH OF AREA
         ST    R15,Request_DataL       SET LENGTH OF AREA
         ST    R15,Response_DataL      SET LENGTH OF AREA
         LA    R15,Request_Area        LOAD ADDRESS OF AREA
         ST    R15,Request_Data@       SAVE ADDRESS OF AREA
         LA    R15,Response_Area       LOAD ADDRESS OF AREA
         ST    R15,Response_Data@      SAVE ADDRESS OF AREA
         SR    R15,R15                 SET TO ZEROES
         ST    R15,Rc                  SET Rc TO ZEROES
         ST    R15,Rsn                 SET RSN TO ZEROES
         ST    R15,Rv                  SET RV TO ZEROES
         ST    R15,Wait                SET Wait TO ZEROES
         EJECT ,
```

```
******** CALL BBOA1INV TO INVOKE EJB
******** (THE SERVICE NAME IS THE TARGET EJB JNDI HOME NAME - WAS IS
******** LOOKING FOR A METHOD CALLED EXECUTE() - THAT WAS WILL INVOKE)
         SPACE 1
         CALL  BBOA1INV,(Register_Name,                                *
               Request_Type,                                           *
               Service_Name,                                           *
               Service_Namelen,                                        *
               Request_Data@,                                          *
               Request_DataL,                                          *
               Response_Data@,                                         *
               Response_DataL,                                         *
               Wait,                                                   *
               Rc,                                                     *
               Rsn,                                                    *
               Rv),VL,MF=(E,APIPlist)
         SPACE 1
         LTR   R15,R15                 ALCS RETURN CODE IS ZEROES
         BNZ   BADBAD                  BRANCH IF NOT - ERROR
         SPACE 1
         L     R0,Rsn                  LOAD REASON CODE
         L     R15,Rc                  LOAD RETURN CODE
         LTR   R15,R15                 INVOKE OK
         BZ    TEST020                 BRANCH IF YES
         SPACE 1
         SERRC E,BAD002                INVOKE FAILED                   *
                                       CHECK R15 AND R00
         EJECT ,
******** TRANSLATE RESPONSE TO EBCDIC
         EJECT ,
TEST020  DC    0H'0'
         LA    R14,256                 LOAD LENGTH OF AREA
         SPACE 1
         ASCIC FROM,DATA=Response_Area,LENGTH=(R14) TRANSLATE TO EBCDIC
         EJECT ,
******** SET UP PARAMETERS AND CALL BBOA1URG TO UNREGISTER
         EJECT ,
         SPACE 1
         XC    fflag,fflag             SET FFLAG
         SPACE 1
         CALL  BBOA1URG,(Register_name,fflag,                          *
               Rc,                                                     *
               Rsn),VL,MF=(E,APIPlist)
         SPACE 1
         LTR   R15,R15                 ALCS RETURN CODE IS ZEROES
         BNZ   BADBAD                  BRANCH IF NOT - ERROR
         SPACE 1
         L     R0,Rsn                  LOAD REASON CODE
         L     R15,Rc                  LOAD RETURN CODE
         LTR   R15,R15                 UNREGISTER OK
         BZ    TEST030                 BRANCH IF YES
         SPACE 1
         SERRC E,BAD003                UNREGISTER FAILED               *
                                       CHECK R15 AND R00
         EJECT ,
******** SEND RESPONSE AND TERMINATE
         EJECT ,
```

```
TEST030  DC    0H'0'
         MVC   EBROUT(3),CE1EID+1      CHANGE EBROUT
         LA    R02,MSG1                ADDRESS OF MESSAGE
         WTOPC TEXTA=(R02),HEADER=NO   SEND RESPONSE
         SPACE 1
         EXITC ,                       TERMINATE
         EJECT ,
******** NON ZERO RETURN CODE FROM ALCS (WAS NOT ENABLED/NOT CONNECTED)
         SPACE 1
BADBAD   SERRC E,BAD000                CHECK REGISTER R15
         EJECT ,
******** CONSTANTS
         SPACE 1
MSG1     DC    AL1(19)                 LENGTH OF RESPONSE TO ORIGINATOR
         DC    CL20'*** TEST OK ***'   TEXT
         EJECT ,
         DS    0D                      WAS OLA CONSTANTS
DGN      DC    Xl8'E2E8F10000000000'   SY1
NODE     DC    Cl8'SY1     '           SY1
SRVR     DC    Cl8'BBOS001 '           BBOS001
RGN      DC    Cl16'HSCBA012        '  HSCBA002
SRVNAME  DC    Cl38'ejb/com/ibm/websphere/hslc/ExecuteHome'
         LTORG ,
         EJECT ,
******** WORK AREA
         SPACE 1
MALOC_AREA    DSECT ,
Daemon_Name   DS CL8                    Addressed by register R02
Server_Name   DS CL8
Node_Name     DS CL8
              DS CL8
Register_Name DS CL16
              DS 0F
Minconn       DS F
Maxconn       DS F
              DS F
Sec           DS F
Request_Type  DS F
Rc            DS F
Rsn           DS F
Rv            DS F
Wait          DS F
fflag         DS F
APIPlist      DS 0F
              DS 20A
Request_Data@ DS A
Request_DataL DS F
Response_Data@ DS A
Response_DataL DS F
Service_Namelen DS F
Service_Name   DS CL256
Service_Name1  DS F
Request_Area   DS CL256
Response_Area  DS CL256
         SPACE 1
MALOCSZ  EQU   *-MALOC_AREA
         SPACE 1
```

```
RSECT ,
SPACE 1
FINIS ,
SPACE 1
END ,
```