



| IBM Software Group

## **64bit support**

### ***Additional topics***



@business on demand.

© 2007 IBM Corporation  
Updated May 14, 2015

This presentation will address any additional 64-bit topics that have not already been covered for a WebSphere Base Application Server version 6.1 on z/OS.

## Agenda

- Recode of C routines in PLX
- BBO\_ECB
- IPCS updates
- Stack and queue class
- WebSphere messaging TCPIP buffer copy in 64bit PLX
- ACRW handling

This presentation will discuss the recoding of C routines in PLX and WebSphere Messaging TCPIP buffer copy in PLX. BBO\_ECB, IPCS updates and ACRW handling will also be discussed.

## C routines rewritten in PLX

SMF utility routines moved from bbooutlc.c:

- GETSTOR, FREESTOR
  - ▶ Redesigned to allocate storage from a CPOOL
    - CPOOL allocated in bboosmf.plx
    - CPOOL ID stored in bacb\_smf\_cpool
- CVC (now in bboosmut.mac)
- LONGAVG (now in bboosmut.mac)
- StrToBin (now in bbooase3.plx)



Several C utility routines for SMF have been replaced with PLX code.

The GETSTOR and FREESTOR functionality have been replaced by PLX code that allocates storage using a CPOOL. The CPOOL is created by bboosmf.plx during SMF initialization and the CPOOL ID is stored in bacb\_smf\_cpool.

The CVC and LONGAVG routines are now in bboosmut.mac, also StrToBin now resides in bbooase3.plx.

## BBO\_ECB – Performance enhancement

- ORB\_Request constructor no longer pre-allocates ECB when initializing the BBO\_ECB object
  - ▶ Noticeable throughput improvement
    - ECB must be below the bar
    - RAS\_MALLOCC31 is slow
  - ▶ The pre-allocated ECB was never used anyway!



The ORB\_Request constructor in bbooorbr.plx was changed to create the BBO\_ECB object without a pre-allocated ECB. Since ECBs must be below the bar, the pre-allocation required RAS\_MALLOCC31, which is extremely slow. The pre-allocated ECB has not been used in the past.

## IPCS changes

- Formatters are in the SBBOMIG dataset
  - ▶ ARE level specific
  - ▶ Must have the SBBOMIG dataset in the IPCS concatenation
- CBDDATA command now BBORDATA. E.g.
  - ▶ Was: IP VERBX **CBDDATA** 'asid(xxxx) config'
  - ▶ Now: IP VERBX **BBORDATA** 'asid(xxxx) config'



IPCS formatters for z/Websphere related dumps have been updated to handle 64bit data. Formatter utility code resides in SBBOMIG and tends to be specific to the version of the code that produced the dump.

The CBDDATA verb to format z/Websphere data is now BBORDATA.

## IPCS changes...

- IPCS commands accept 64-bit addresses
  - ▶ 12345678\_12345678 or 123456781234578
  - ▶ Leading zeros optional

```
verbx bbordata 'asid(56),objtype(03001000),objaddr(1_08E822E0)'
```

- Formatter detects whether server was in 31-bit or 64-bit
  - ▶ Dump formatted accordingly
  - ▶ BBO3BACB field BACB\_FLAGS\_B1:

```
BACB_FLAGS_B1:  
Addr mode bit was set  
zWAS in 64bit mode
```

IPCS commands now accept 64-bit addresses, which can be entered with or without an underscore separator before the last 8 digits. Leading zeros are optional. Here is an example of a command to format the SessionManager object which is above the bar.

The formatter determines whether the dump was 64-bit or 31-bit mode by looking at a new flag in the BACB. For 64-bit mode dumps, you will see “zWAS in 64bit mode” when formatting the BACB. The absence of this message means the dump is 31-bit.

## IPCS changes...

For 64-bit dumps, 64-bit addresses are displayed:

- BBORxxxx messages

```
BBOR0028I Formatting BTCB for TCB 00000000_005BC9C0
```

- Control block dump

```
+0258 0rb0bjP.. 00000001 08B47E60
```

- Object formatting

```
m_SessionManagerCleanUpRtn (00000001_08E82350): 00000001_08B4EE60
```



Addresses in BBORxxxx messages will always be formatted as 64-bit, even if they are only 31-bit, for example, TCB.

In a control block dump, a 64-bit address will appear as two words without the \_ connector. In a formatted object, the address in the dump of an attribute is shown with the underscore connector and, if the value of the attribute is a 64-bit pointer, it will as well.

## Stack and queue class

- Standard stack and queue classes serialized using CDS of the next pointer and sequence #.
  - ▶ Doesn't work with 8 byte pointers
  - ▶ CDSG supported in C++ (16 byte CS)
  - ▶ Class templates changed to use it when in 64bit mode



Several stack and queue classes depend on compare and swapping for the serialization when adding or deleting member elements. The next element address and the sequence number are what get tested and swapped using a CDS. This had to be changed for 64bit because the 8 byte pointer maxed out the CDS. Since CDSG is supported natively in the C/C++ compiler, classes that use a pointer plus sequence number for serialization were updated.



## Stack and queue class

- Header definition

```
struct {  
    void      *      first_word;  
    unsigned long second_word;  
} double_word;
```

- Serialization scheme

```
while (COMPARE_AND_SWAP_PTR_SEQ(  
    &oldCompareArea,  
    stack_HeaderPtr,  
    newCompareArea )  
    );
```

The first update is in the stack or queue element header. The design was to allow for an 8 byte compare and swap in 31-bit mode and 16 bytes in 64-bit mode. By making a long of the sequence number, or whatever the low order component is, the header is the right size in both build modes.

Other changes included updating the means for doing the compare and swap. It expands into a CDS or CDSG depending on the compile mode.



## WebSphere messaging TCPIP buffer copy in 64bit PLX

### ▪ Background

- ▶ Buffers start out in JAVA code and get passed through C to PLX using glue code.
- ▶ PLX copies data buffers to/from a dataspace.
- ▶ Performance fix to eliminate glue

### ▪ Updates

- ▶ Send and receive glue routine do not copy data buffer
- ▶ Pass buffer address as long long



An enhancement was made to the CRA WebSphere Messaging buffer handling. These buffers contain data that was copied into or out of a z/OS data space. C code now deals with the data in buffers and there is an interface to the PLX code that handles the copying with the data space.

In 64-bit mode, the interface has to be through glue code. The primary function is to copy the buffers below the bar when the data is on the way in, and above the bar when the data is on the way back out. Since the buffers are big chunks of data, the glue copying takes time.

It turns out that the PLX code itself does not do much but copy the buffer. Given this, it was too much overhead to copy the data first in the glue code and very soon after copy it a second time in the PLX code.

The buffer handling was changed to eliminate the extra copying by making the PLX code smart enough to do it in 64-bit mode. The glue code was changed to not do the buffer copy. The glue code is still necessary, as the plist and parameters are needed to be below the bar because the PLX module itself is still in 31-bit mode. That also meant that the buffer address could not be a 64 pointer. The PLX compiler does not accept declaring ptr(64) in 31-bit mode so a fixed(64) or in C long is used.

## WebSphere messaging TCPIP buffer copy in 64bit PLX...

- PLX code: JfapBufferCopyTo / From
- Declare 64bit code block `Begin amode(64) ;`
- Copy the input buffer long long pointer to actual pointer
- Issue `?SYSSTATE AMODE64(YES / NO)` before and after Move with Key macros

In the receive and send buffer PLX modules, a 64-bit aware block of code is used by the `begin Amode(64)` construct. Within that code the compiler goes into 64-bit mode and allows specifying 64-bit pointers. The `SYSSTATE` macro enables the generated macro code to be 64-bit aware. For example, register operands are treated as 64-bit registers.

## ACRW handling

- ACRW passes data and pointers in its parameter fields.
  - ▶ Fields re-arranged
  - ▶ Additional pointers and mpointers included

```

struct _mptr_struct {
    __mptr(void, _acrw_mptr1);
    __mptr(void, _acrw_mptr2);
    __mptr(void, _acrw_mptr3);
    __mptr(void, _acrw_mptr4);
};
struct _dword_struct {
    long long _acrw_datadw1;
    long long _acrw_datadw2;
    long long _acrw_datadw3;
    long long _acrw_datadw4;
};
struct _word_struct {
    int _acrw_data1;
    int _acrw_data2;
    int _acrw_data3;
    int _acrw_data4;
    int _acrw_data5;
    int _acrw_data6;
    int _acrw_data7;
    int _acrw_data8;
};

union {
    unsigned char _acrw_data[32];
    struct _mptr_struct _mptrs;
    struct _dword_struct _dwords;
    struct _word_struct _words;
} _acrw_u_data;

```

The last topic to cover is the ACRW. The ACRW which is a general purpose passer of data and pointer information, was affected by 64-bit support. The fields in the ACRW were rearranged and now can accommodate four 8 byte pointers or mptrs and 8 one word data areas. The areas are overlaid and so care needs to be taken when passing and extracting information in the ACRW.

## Trademarks, copyrights, and disclaimers

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

IBM                      WebSphere

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2007. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

