



# DIG64 Processor Abstraction Layer Overwrite (PALO) Table Specification R1.0

A publication of the Consortium of  
Developers' Interface Guides for Intel® Itanium®  
Architecture-based Servers (DIG64)

December 2007

---

Bull SAS  
Fujitsu Siemens Computers  
Hewlett-Packard Company  
Hitachi Limited  
Intel Corporation  
NEC Corporation  
Unisys Corporation

Notice: Implementations developed using the information provided in this specification may infringe the intellectual property rights of various parties including, but not limited to, the Promoters and parties involved in the development of this specification. An agreement to grant a limited patent license is available under the terms of a DIG64 Adopters Agreement, a copy of which may be obtained from <http://www.dig64.org>.

THIS DOCUMENT IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. The Promoters of this document disclaim all liability, including liability for infringement of any proprietary rights, relating to use of information in this specification. No license, express or implied, by estoppel or otherwise, to any intellectual property rights (including without limitation rights under any party's patents) is granted herein, except that a copyright license is hereby granted to you to copy and reproduce this specification for your internal use only.

Copyright © 2007 Bull SAS, Fujitsu Siemens Computers, Hewlett-Packard Company, Hitachi Limited, Intel Corporation, NEC Corporation, and Unisys Corporation (collectively referred to as the "DIG64 Promoters"). All rights reserved.

\*Other product and corporate names may be trademarks of other companies and are used only for explanation and to the owners' benefit, without intent to infringe.

# Section 1 Introduction

---

The Processor Abstraction Layer (PAL) does not have platform knowledge. This is a specification for a new UEFI Configuration Table to report platform capabilities which may be different than what is reported by PAL. This table is used to report a different maximum number of concurrent global TLB purges (from the *ptc.g* or *ptc.ga* instruction) than what is reported by PAL\_VM\_SUMMARY.

## Section 2 Definition

---

### 2.1 Processor Abstraction Layer Overwrite Table

---

This specification declares a new *EFI Configuration Table* that reports Itanium platform capabilities that may be different from what is reported by the Itanium processor. Its location and Globally Unique Identifier (GUID) pair are obtained from the *EFI System Table* (refer to the *Unified Extensible Firmware Interface Specification*, available from <http://www.uefi.org>).

*Summary:*

The purpose of the Processor Abstraction Layer Overwrite (PALO) Table is to report platform capabilities that may be different from what is reported by the Itanium processor. This table is optional. Platforms that support the exact capabilities as reported by the Itanium Processor Abstraction Layer (PAL), are not required to implement it. The PALO takes precedence over the capabilities reported by PAL. The PALO is static. It does not change during the OS Runtime. System Firmware is the producer of the table and it is read-only from the OS point of view.

*GUID:*

```
#define PROCESSOR_ABSTRACTION_LAYER_OVERWRITE_GUID \  
{0x6cb0a200,0x893a,0x11da,0x96,0xd2,0x00,0x10,0x83,0xff,0xca,0x4d}
```

The address reported in *VendorTable* in the *EFI\_CONFIGURATION\_TABLE* structure (refer to the *Unified Extensible Firmware Interface Specification*, available from <http://www.uefi.org>) is always physical.

**Table Layout:**

Field	Offset (Bytes)	Length (Bytes)	Description
SIGNATURE	0	4	The ASCII representation of "PALO", which confirms the presence of the table.
LENGTH	4	4	The length of this entire table in bytes, starting from offset zero and including the header and all entries (24).
MINOR REVISION	8	1	Minor revision of this table, coded as 0x00.
MAJOR REVISION	9	1	Major revision of this table coded as 0x02.

DIG64 Processor Abstraction Layer Overwrite Table Specification  
Release 1.0

CHECKSUM	10	1	A modulo checksum of the entire table, starting from offset zero and including the header and all entries. All bytes, including the CHECKSUM, must add up to zero.
RESERVED	11	5	Reserved. Must be zero.
MAX_TLB_PURGES	16	2	16-bit unsigned integer which designates the maximum number of concurrent outstanding global TLB purges allowed by the platform. This value may be less than what is reported by PAL_VM_SUMMARY. When MAX_TLB_PURGES is equal to 0, it means that the platform does not support any global TLB purges. In addition, the value $2^{16}-1$ specifies that there is no limits on multiple concurrent outstanding global TLB purges.
Reserved	18	6	Must be set to zero. Reserved for future use.

## 2.2 Example

---

This section provides an example of implementing the PALO table. It is only for informative purpose.

```
#include EFI_GUID_DEFINITION(palo)

EFI_GUID_STRING(&gEfiPaloTableGuid, "PALO Table", "PALO Table GUID in
EFI System Table");

#define EFI_PALO_TABLE_GUID \
    { 0x6cb0a200, 0x893a, 0x11da, {0x96, 0xd2, 0x00, 0x10, 0x83, 0xff,
    0xca, 0x4d }}

EFI_GUID gEfiPaloTableGuid = EFI_PALO_TABLE_GUID;

typedef struct
{
    CHAR8    Signature[4];
    UINT32   Length;
    UINT8    Minor_Revision;
    UINT8    Major_Revision;
    UINT8    Checksum;
    UINT8    Reserved_one[5];
    UINT16   MAX_TLB_PURGES;
    UINT8    Reserved_two[6];
} PALO_Structure_Table;

UINT8
Compute_Checksum(UINT8 *address_, UINT32 length_)
{
    UINT8    sum = 0;
    UINT8    checksum;
    UINT64   i;

    for(i = 0; i < length_; i++)
    {
        sum = sum + address_[i];
    }
    checksum = (UINT8)(0 - sum);
    return checksum;
}

PALO_Structure_Table *palo_table;

palo_table = (PALO_Structure_Table *)PALOAddress;
palo_table->Signature[0] = 'P';
```

```
palo_table->Signature[1] = 'A';
palo_table->Signature[2] = 'L';
palo_table->Signature[3] = 'O';
palo_table->Length       = 24;
palo_table->Minor_Revision = 0x00;
palo_table->Major_Revision = 0x02;
palo_table->Checksum      = 0;
palo_table->MAX_TLB_PURGES = 4;

// Zero out bits reserved for future use
for(i = 0; i < 5; i++)
{
    palo_table->Reserved_one[i] = 0;
}

// Zero out bits reserved for future use
for(i = 0; i < 6; i++)
{
    palo_table->Reserved_two[i] = 0;
}

palo_table->Checksum = Compute_Checksum((UINT8 *)PALOAddress,
                                       palo_table->Length);

BS->InstallConfigurationTable(&gEfiPaloTableGuid, palo_table);
```

Resulting table in memory is:

```
50 41 4c 4f  18 00 00 00      PALO....
00 02 b6 00  00 00 00 00      .....
04 00 00 00  00 00 00 00      .....
```