Live Update Handover Protocol

David Woodhouse <dwmw@amazon.co.uk>

Julien Grall jgrall@amazon.com

Contents

% Revision 0.1

1 Introduction

1.1 Purpose

Live update performs a *kexec* from one running version of Xen to another, preserving all running domains in a form of guest-transparent live migration.

This document outlines the memory layout requirements and data stream used in handover protocol, to ensure that pages used by running domains are preserved during the transition from one version of Xen to the next.

1.2 Compatibility

It cannot be repeated often enough that information passed over live update is an ABI. It is expected that live update can be performed from one major version of Xen to another, or even hypothetically to a system which is not Xen at all.

It is necessary that some data are handed over "in place"; in particular the memory pages of the running domains. However, no internal Xen data structures may be transferred in this fashion; at least not without retrospectively declaring them to be ABI, with the restrictions that places on subsequent changes.

2 Handover

2.1 Memory Usage Restrictions

The new Xen must take care not to use any memory pages which already belong to domains. To facilitate this, multiple regions of memory are reserved for the boot allocator, known as *live update bootmem*.

One of the regions must be reserved in memory just below Xen to allow the binary to grow. This region is reserved by the original Xen during its own boot, and the location is made available to the *kexec(8)* user space tool through the `kexec_get_range` hypercall using a new region type `KEXEC_RANGE_MA_LIVEUPDATE`. It is passed to the new Xen on the command line, using the `liveupdate=` parameter.

The extra regions will be described in the *Live Update Data Stream*.

The new Xen must not use any pages outside those regions until it has consumed the live update data stream and determined which pages are already in use by running domains.

At run time, Xen may use memory from the reserved regions for any purpose that does not require preservation over a live update; in particular it must not be shared (or mapped) to a domain.

Since the current region used by Xen is known to be available, the new Xen executable image must be loaded by kexec below the end address of the existing Xen. For that reason, freed init memory from the Xen image is also treated as reserved *live update bootmem*.

2.2 Live Update Data Stream

During handover, the running Xen pauses all domains and creates a *live update data stream* containing all the information required by the new Xen to restore them. This is largely the same as guest transparent live migration.

Data pages for this stream may be allocated anywhere in physical memory outside the *live update bootmem* regions.

Xen creates an array of MFNs of the allocated data pages, suitable for passing to `vmap()` to obtain a virtually contiguous mapping of the whole data stream. The MFN array is physically contiguous, but the MFNs do not have to be.

2.3 Breadcrumb

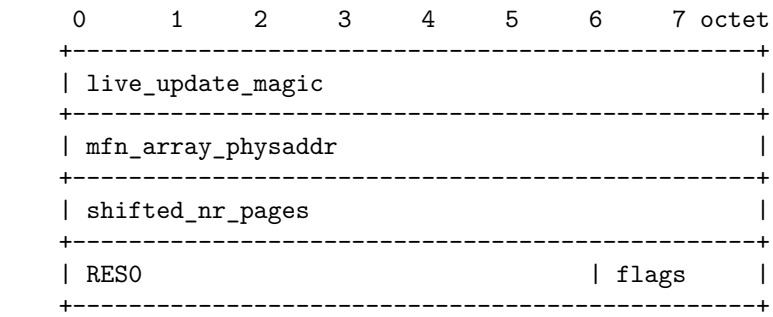
Since the live update data stream is created during the final `kexec_exec` hypercall, its address cannot be passed on the command line to the new Xen since the command line needs to have been set up by `kexec(8)` in userspace long beforehand.

Thus, to allow the new Xen to find the data stream, the old Xen places a *breadcrumb* in the first words of the *live update bootmem*, containing the number of data pages, and the physical address of the contiguous MFN array.

The breadcrumb is written as the last action of the `kexec_reloc()` routine during the `kexec` handover, so cannot overwrite anything important by virtue of the existing guarantee that Xen will not place any data in that region which needs to survive across a live update.

A restriction of the `kexec_reloc()` mechanism for writing the breadcrumb is that the values are host-endian and are masked with PAGE_MASK; the low bits are zeroed. This is actually perfect for the magic value used to recognise a live update breadcrumb, since it neatly prevents any attempt to live update to a Xen which uses a different endianness or page size.

For the physical address of the MFN list it's perfectly fine, since that list is page-aligned anyway. For the number of pages, it means the value must be shifted accordingly. Hence the use of `shifted_nr_pages` in the breadcrumb structure below:



Field	Description
live_update_magic	"LiveUpda" (0x4c69766555706461) stored in the the host endianness and masked with PAGE_MASK. For example on x86_64: `00 60 70 55 65 76 89 4c`.

Field	Description
mfn_array_physaddr	Machine address of MFN list for data stream. Each item is an 8-byte MFN.
shift_nr_pages	Number of data pages, shifted by PAGE_SHIFT to avoid the limitation of kexec_reloc().
flags	Bit 0: Per-record stats enabled (see record definition) Bit 1 - 15: RES0

2.4 IOMMU

Where devices are passed through to domains, it may not be possible to quiesce those devices for the purpose of performing the update.

If performing live update with assigned devices, the original Xen will leave the IOMMU mappings active during the handover (thus implying that IOMMU page tables may not be allocated in the `live update bootmem` region either). Moreover, no part of the reserved memory should be exposed via IOMMU mappings.

The new Xen must resume control of the IOMMU without causing those mappings to become invalid even for a short period of time. On hardware which does not support Posted Interrupts, interrupts may need to be generated on resume.

This section will be expanded once we actually have it working.

3 Data Stream Overview

Once discovered and mapped, the live update data stream forms a virtually contiguous stream of records.

Some record types from the libxenctrl (libxc) Domain Image Format (see `docs/specs/libxc-migration-stream.pandoc`) are used as-is, such as the `X86_PV_INFO`, `X86_PV_VCPU_BASIC`, `HVM_CONTEXT` and certain other records containing domain-specific data. The ones used will be listed below.

Other new record types specific to the live update process are defined in this document. Of those, some contain global state such as the M2P table information, while others are domain-specific.

The live update data stream starts with records containing global information, followed any number of times by a `LU_DOMAIN_INFO` record and subsequent domain-specific records for that domain.

There is a single `END` record at the end of the live update data stream, indicating that no more `LU_DOMAIN_INFO` records are present.

4 Fields

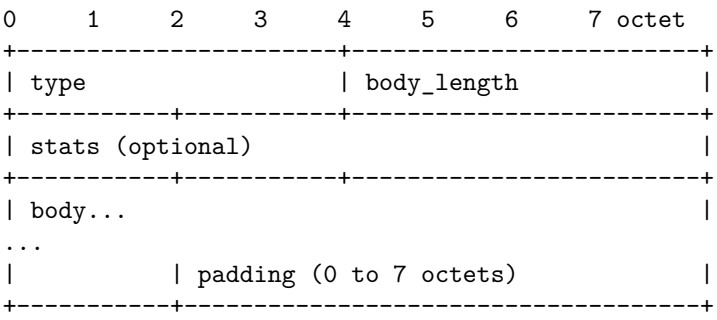
All the fields within the records have a fixed width.

Fields are always aligned to their size.

Padding and reserved fields (RES0) are set to zero on save and must be ignored during restore.

Integer fields in the records are host-endian.

As defined in the LibXenCtrl Domain Image format document, a record has the following structure. Record type values defined for live update have bit 30 set, and are thus in the range 0x40000000-0x7FFFFFFF for mandatory live update records, and 0xC0000000-0xFFFFFFFF for optional live update records *(of which there are none at the present time)*.



Field	Description
type	0x40000000: LU_VERSION 0x40000001: LU_DOMAIN_INFO 0x40000002: FREEMEM_INFO 0x40000003: M2P_LIST 0x40000004: COMPAT_M2P_LIST 0x40000005: LU_X86_TSC_INFO 0x40000006: LU_GLOBAL_INFO 0x40000007: LU_TIMESTAMP 0x40000008 - 0x40000011: Reserved for future *mandatory* live update records. 0x40000012: PIRQ_INFOS 0x40000013: LU_PAGE_INFOS 0x40000014: VCPU_INFO 0x40000015: PIRQ_EOI 0x40000016: P2M_INFO 0x40000017: HAP_INFO 0x40000018: LU_X86_E820 0x40000019: VLAPIC_MAPPING 0x4000001A - 0x4000001A: Reserved for future *mandatory* live update records. 0x4000001B: CLOCK 0x4000001C: VCPU_TIMER_PERIODIC 0x4000001D: VCPU_TIMER_SINGLESHOT 0x4000001E: GRANT_TABLE 0x4000001F: GRANT_MAPPINGS 0x40000020: EVTCHN_FIFO_CONTROL_BLOCK 0x40000021: EVTCHN_FIFO_ARRAY

Field	Description
	0x40000022 - 0x40000022: Reserved for future *mandatory* live update records. 0x40000023: PCI_DEVICES 0x40000024: VCPU_AFFINITY 0x40000025: VCPU_RUNSTATE 0x40000026 - 0x40000028: Reserved for future *mandatory* live update records. 0x40000029: X86_RTC_INFO 0x4000002A: KDUMP_INFO 0x4000002B: KDUMP_IMAGE 0x4000002C - 0x4000002C: Reserved for future *mandatory* live update records. 0x4000002D: DOM_IOMMU_INFO 0x4000002E: SYS_IOMMU_INFO 0x4000002F: CPUID_INFO 0x40000030: IOSERV_INFO 0x40000031: IOSERV_VCPU 0x40000032: IOSERV_RANGES 0x40000033: X86_HVM_PT_PIRQS 0x40000034: SYS_VPMU_INFO 0x40000035: HVM_VPMU_CONTEXT 0x40000036 - 0x7FFFFFFF: Reserved for future *mandatory* live update records. 0xC0000000 - 0xFFFFFFFF: Reserved for future *optional* live update records.
body_length	Length in octets of the record body.
stats	Per-record stats when enabled in the breadcrumb. If disabled, the field is 0-length.
body	Content of the record.
padding	0 to 7 octets of zeros to pad the whole record to a multiple of 8 octets.

The following records are defined in the LibXenCtrl Domain Image format document and re-used by LiveUpdate: * 0x00000000 : END * 0x00000004 : X86_PV_VCPU_BASIC * 0x00000005 : X86_PV_VCPU_EXTENDED * 0x00000006 : X86_PV_VCPU_XSAVE * 0x00000009 : HVM_CONTEXT * 0x0000000A : HVM_PARAMS * 0x0000000C : X86_PV_VCPU_MSRS

When per-record stats is enabled (set in the breadcrum), each record will contain an extra 16-bytes before the body that will be interpreted as the following:

```
 0      1     2     3     4     5     6     7 octet
+-----------+-----------+-----------+-------------+
| timestamp_open                                   |
```

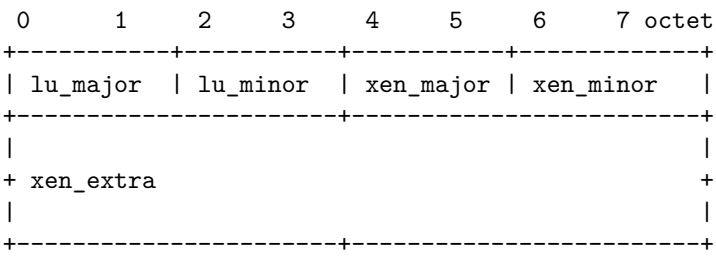
```
+-----------+-----------+-----------+-------------+
| timestamp_close                                 |
+-----------+-----------+-----------+-------------+
```

Field	Description
timestamp_open	Time Stamp when the record is opened
timestamp_close	Time Stamp when the record is closed. This can be 0 if the work done betwen the open/close is meaningless (e.g. the record was just appended).

5 Global Records

5.1 LU_VERSION

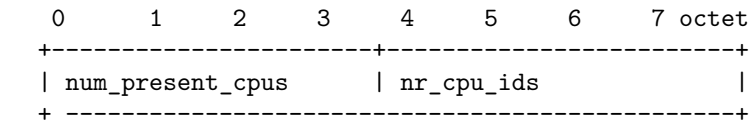
The version field indicates the version of Xen from which the system is live updating. In theory this should never be relevant, but it allows for version-specific workarounds to be implemented in the receiving Xen should they become necessary.



Field	Description
lu_major	The major version of the stream format (currently 0)
lu_minor	The minor version of the stream format (currently 1)
xen_major	The Xen major version from which the system is updating.
xen_minor	The Xen minor version from which the system is updating.
xen_extra	The Xen extra version from wich the system is updating. This is a NUL-terminated string. Unused bytes must be NUL.

5.2 LU_GLOBAL_INFO

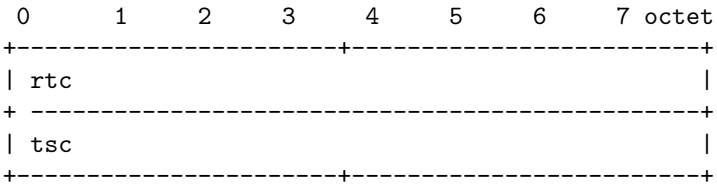
This record describes global information, either for sanity checks or for the next Xen to restore global fields.



Field	Description
num_present_cpus	Total number of CPUs present in the system.
nr_cpu_ids	Number of CPUs Xen is allowed to bring up. Usually the same as num_present_cpus, but not always (e.g. when "maxcpus=" is specified on the Xen command line on x86).

5.3 X86_RTC_INFO

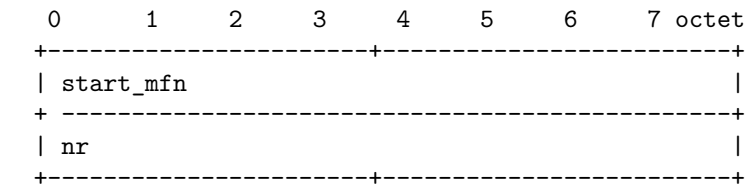
RTC information the next Xen should be initialised with.



Field	Description
rtc	The RTC value (seconds since Epoch).
tsc	The TSC timestamp when the RTC value was sampled.

5.4 FREEMEM_INFO

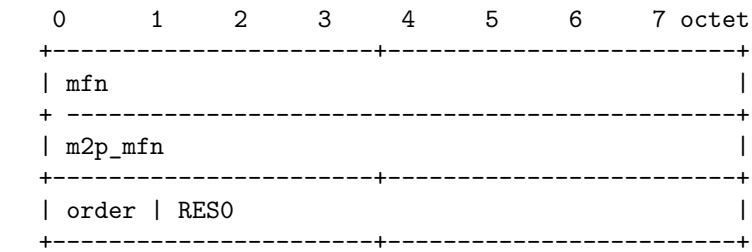
This record contains free memory chunks that can be safely used by the next Xen after LU during early boot. Such chunks must be memory that the previous Xen does not need to preserve across an LU. This record may contain an arbitrary number of entries, each describing a free memory chunk. Below is the layout of a single entry.



Field	Description
start_mfn	The start MFN of this entry.
nr	Number of pages that can be used from start_mfn.

5.5 (x86) (COMPAT_)M2P_LIST

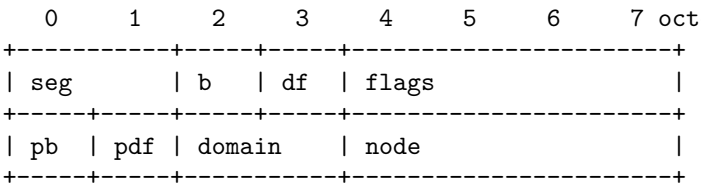
LU needs to preserve the M2P since it is visible to PV guests. This record contains a list of entries that describe all the M2P chunks in memory. Below is the format of one M2P entry. Note that COMPAT_M2P_LIST and M2P_LIST share the same format.



Field	Description
mfn	The start MFN this M2P chunk corresponds to.
m2p_mfn	The MFN where this chunk is.
order	The order of this M2P chunk in terms of pages.

5.6 PCI_DEVICES

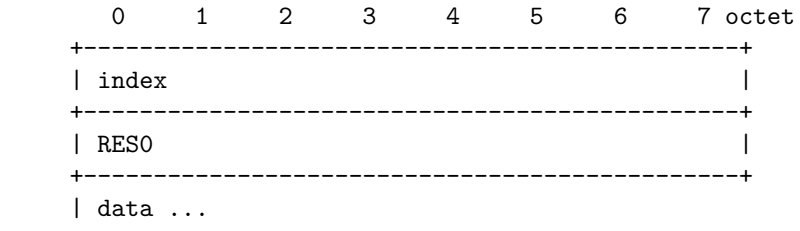
This record is an array containing an entry for each PCI device in the system. After LU, the next Xen will use this array to build its device list rather than re-scanning the bus. Each element of the array has the following format.



Field	Description
seg	Device 'segment'.
b	Device 'bus'.
df	Device 'devfn'.
flags	bit 0 - bit 4: Correspond to those used in PHYSDEVOP_pci_device_add. bit 31: Was the MSI-X prepared?
pb	Physical device 'bus' (if the device is a VF).
pdf	Physical device 'devfn' (if the device is a VF).
domain	Domain ID of the owner
node	The NUMA node to which the device is attached.

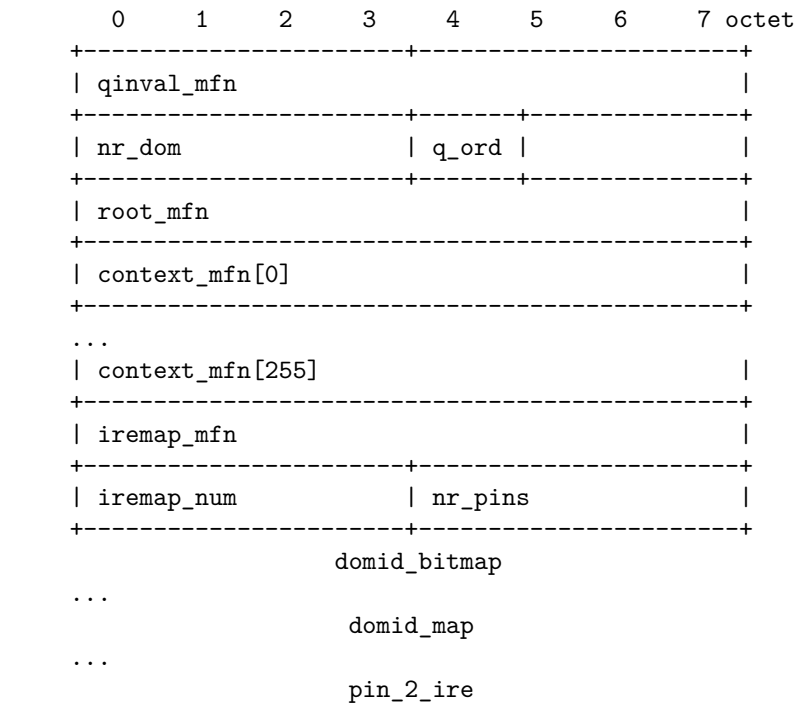
5.7 SYS_IOMMU_INFO

Global system IOMMU state.



Field	Description
index	An identifier, used by vendor specific code, to identify the IOMMU.
data	Vendor-specific IOMMU state.

Currently there is only support for VT-d (Intel IOMMU) state.

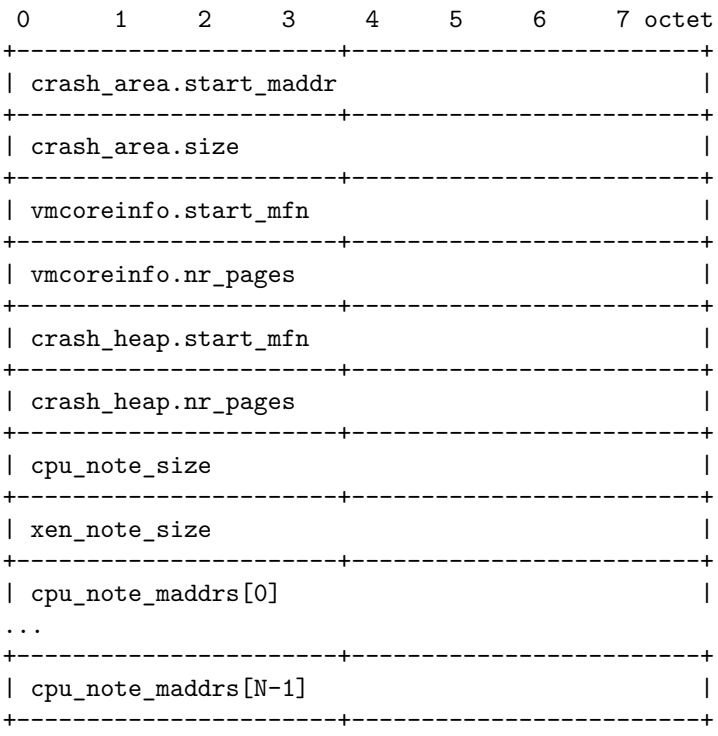


Field	Description
qinval_mfn	The base MFN of the region used for queued invalidation.

Field	Description
nr_dom	Number of domain id values supported by the h/w. This determines the size of domid_bitmap and domid_map.
q_ord	Order of the invalidation queue size + 1.
root_mfn	The root table used for DMAR.
context_mfn[]	The array of 256 context entries matching all entries in the root table (INVALID_MFN represents a non-present entry).
iremap_mfn	The base MFN of the interrupt remapping table
iremap_num	The number of allocated interrupt remapping entries
nr_pins	The total number of pins in all IOAPIC units. This determines the size of pin_2_ire.
domid_bitmap	Bitmap used for allocating entries in domid_map.
domid_map	A mapping from domain id to (Xen) domid.
pin_2_ire	A mapping from IOAPIC pin to interrupt remapping entry index.

5.8 KDUMP_INFO

This record describes basic kdump state, so as to allow the next Xen to reuse the kdump infrastructure set up by the previous Xen. It is present only if kdump is enabled for the previous Xen (i.e. via "crashkernel=" on the boot command line).



Field Description

crash_area .start_maddr Machine address of the start of the crash kernel area. .size Size of the crash kernel area, in bytes.

vmcoreinfo .start_mfn MFN of the first vmcoreinfo page. .nr_pages Number of pages allocated for vmcoreinfo.

crash_heap .start_mfn MFN of the first crash heap page. .nr_pages Number of pages allocated for the crash heap.

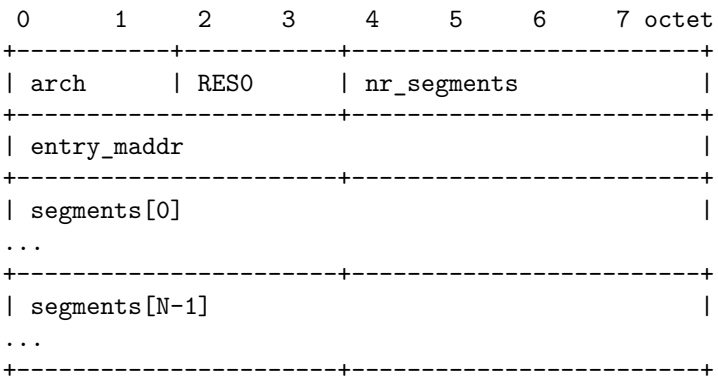
cpu_note_size Size of the per-CPU crash note buffer, where CPU registers will be dump on crash.

xen_note_size Size of the Xen crash note buffer, which contains Xen version information. This buffer is placed at the end of CPU0's crash note buffer.

cpu_note_maddrs[] An array of N = nr_cpu_ids (see LU_GLOBAL_INFO) machine addresses, each pointing to the start of the per-CPU crash note buffer for one CPU. If an address reads -1, it means the previous Xen did not set up the crash note buffer for this CPU.

5.9 KDUMP_IMAGE

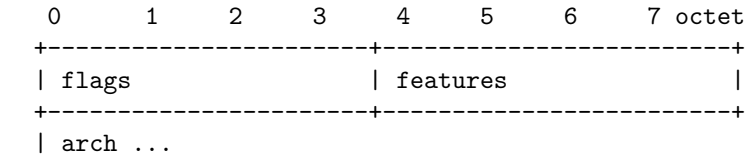
This record describes a loaded kdump (crash kernel) image, which is the only kind of kexec image that can be loaded side by side with a Live Update image. After Live Update, the same kdump image will remain loaded in the next Xen. This record is omitted no kdump image is loaded in the previous Xen.



Field	Description
arch	One of the standard ELF machine types (EM_*) indicating the target architecture for this image.
nr_segments	Number of ELF segments in this image.
entry_maddr	Machine address of the crash kernel entry point. Must be within the target maddr range of one of the ELF segments in this image.
segments[]	An array of N = nr_segments elements describing the ELF segments in this image. This should be a copy of the array of segment descriptors (each of type xen_kexec_segment_t) that was passed to the previous Xen via KEXEC_CMD_kexec_load for loading this image.

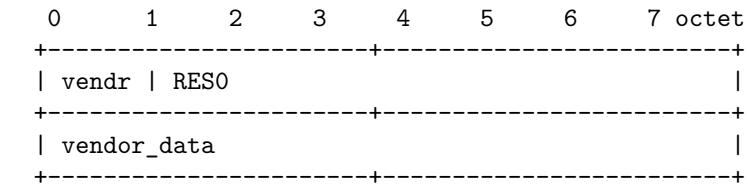
5.10 SYS_VPMU_INFO

This record describes global vPMU state, including system-wide PMU virtualization settings and the host PMU hardware configuration. It is present only if PMU virtualization is enabled for the previous Xen (i.e. via "vpmu=" on the boot command line).



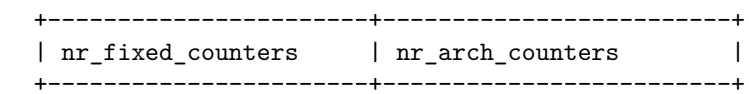
Field	Description
flags	System-wide PMU virtualization flags. * Bit 0: vPMU is exposed to HVM guests only. Implies "vpmu=nopv" and XENPMU_MODE_SELF.
features	PMU features exposed to guests (XENPMU_FEATURE_*).
arch	Architecture-specific global vPMU state (see layout below).

Currently, arch is only defined for x86.



Field	Description
vendr	PMU vendor. Must be one of the following. * 0: Intel. * 1: AMD.
vendor_data	Vendor-specific global vPMU state (see layout below).

Currently, vendor_data is only defined for the Intel PMU.

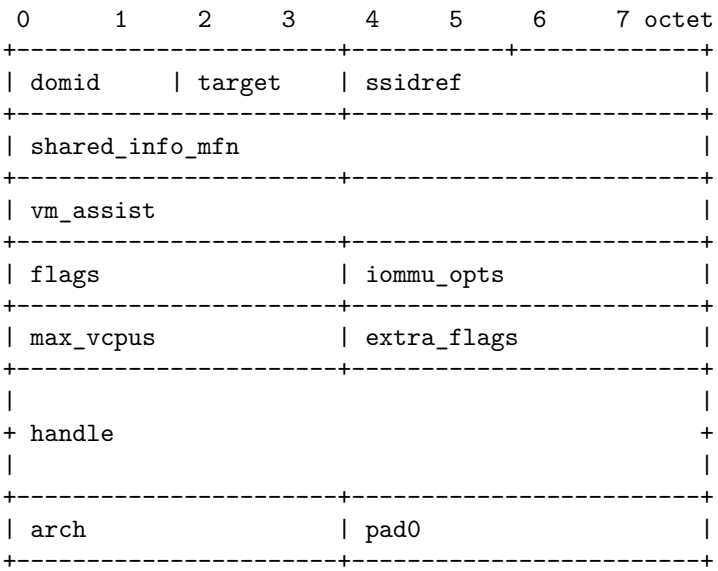


Field	Description
nr_fixed_counters	Number of architecturally-defined, fixed-purpose PMU counters per CPU.
nr_arch_counters	Number of architecturally-defined, general-purpose PMU counters per CPU.

6 Per-domain Records

6.1 LU_DOMAIN_INFO

This record contains basic domain information which is used to reconstruct each domain after LU. It is only enough to recreate the domain and vCPUs.



Field Description

domid Domain ID of the domain being reconstructed.

target The domain ID this domain is allowed to manage.

shared_info_mfn The MFN of the shared info page for this domain.

vm_assist VMASST_TYPE_* bitmap as described in Xen public headers.

flags Attributes used to create the domain. Bit 0-6: LU_DOMCTL_CDF_* in `stream_format.h`

iommu_opts IOMMU-related flags. Bit 0: LU_DOMCTL_IOMMU_no_sharept

max_vcpus Max number of vCPUs for this domain.

extra_flags Extra attributes that may be configured after the domain is created. Bit 0-3: DHDR_FLAGS_* in `stream_format.h`

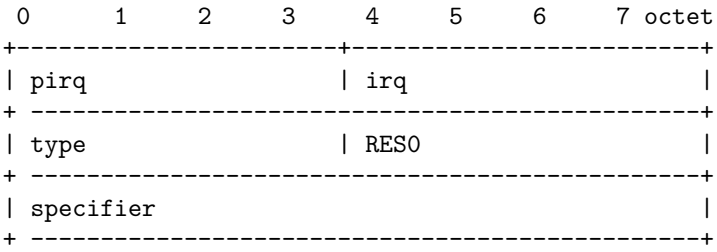
handle Domain handle.

arch Architecture specific info. (x86) emulation_flags Emulation flags. 32 bits. Bit 0-10: LU_X86_EMU_* in `x86/stream_format.h`

6.2 pad0 Padding to 8-byte aligned boundary.

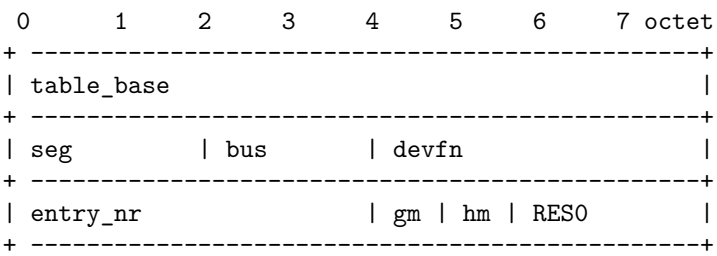
6.3 PIRQ_INFOS

This record describes each PIRQ for a given domain.



Field	Description
pirq	Index of PIRQ in the domain's PIRQ tree.
irq	Index of the host IRQ. The value is unknown if the PIRQ is not mapped to a host IRQ.
type	Type of the PIRQ. Possible values are: 0: MSI 1: GSI 2: Allocated but not yet mapped 3: Emulated IRQ (HVM only)
specifier	Information specific to type

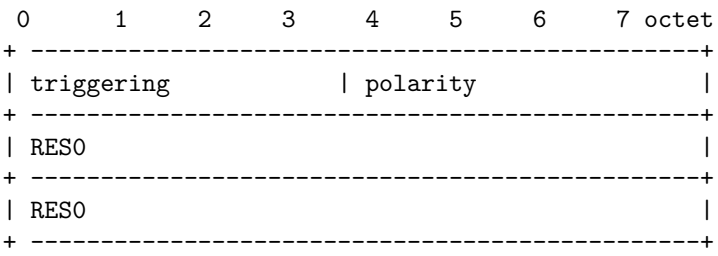
The specifier for type 0 (MSI) is:



Field	Description
table_base	Base address of MSI-X table structure. For MSI, it is invalid and always 0.
seg	PCI segment of the associated device.
bus	PCI bus of the associated device.
devfn	Concatenation of PCI device and function of the associated device.
entry_nr	In case of MSI, entry index of the MSI if the device is multi message capable and uses multiple vectors. In case of MSI-X, index of the MSI-X table entry. Currently this field is always 0 for MSI, as secondary MSIs are not supported and hence the information is not transferred.

Field	Description
hm	Stands for host masked. This field specifies if the MSI descriptor has its host mask field is set. This is related with physical MSI masking.
gm	Stands for guest masked. This field specifies if the MSI descriptor has its guest masked field is set. This is related with virtual MSI (vMSI) masking.

The specifier for type 1 (GSI) is:

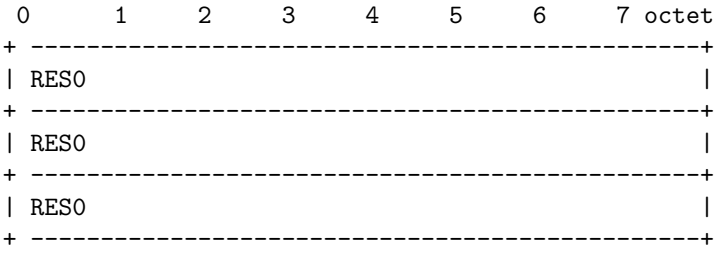


Field Description

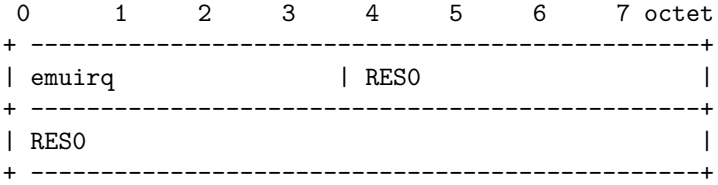
triggering Triggering type of the PIRQ (0 for edge or 1 for level)

6.4 polarity Polarity of the PIRQ (0 for active high or 1 for active low)

The specifier for type 2 (Allocated but not mapped) is:



The specifier for type 3 (Emulated IRQ) is:



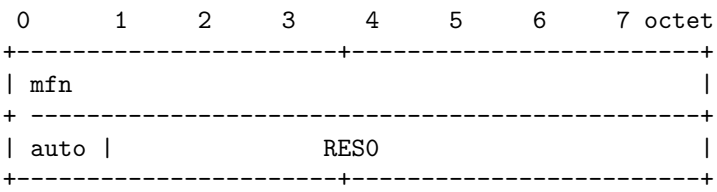
```
| RES0                                                          |
+ --------------------------------------------------------------+
```

Field Description

6.5 emuirq Emulated interrupt number

6.6 PIRQ_EOI

This record contains per-domain MFN of a shared page for end-of-interrupt (EOI) signalling, which is used by Xen to indicate whether the guest must issue PHYSDEVOP_eoi.



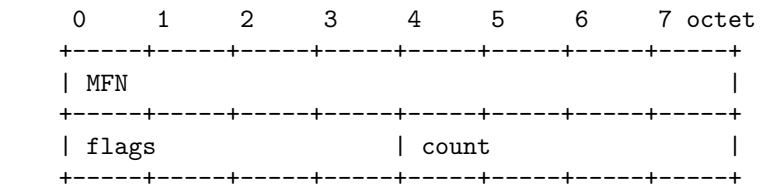
Field	Description
mfn	MFN of shared page.
auto	Indicates whether PHYSDEVOP_eoi automatically unmasks the associated event channel.

6.7 LU_PAGE_INFOS

This record contains the list of MFN owned by a domain.

Field	Description
max_pages	Maximum number of pages the domain can own
page[X]	Information of a given page (see layout below)

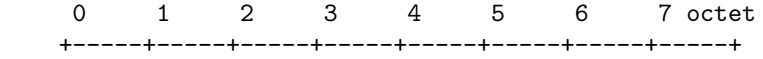
Each page[x] has the following layout:

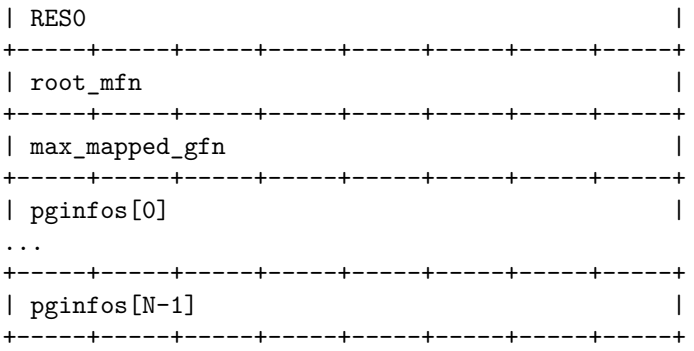


Field	Description
MFN	Machine Frame Number of the page we are preserving
flags	Bitfields [31] Is the page pinned? [30-28] Page type * 0x0: Normal RAM * 0x1: Contains L1 guest page-table * 0x2: Contains L2 guest page-table * 0x3: Contains L3 guest page-table * 0x4: Contains L4 guest page-table * 0x5: Broken page * All the other values are reserved for future use [27-0] Reserve for future use
count	Number of contiguous pages starting at MFN with the same flags

6.8 P2M_INFO

Describes the P2M page table structure.

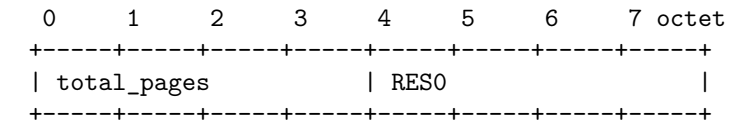




Field	Description
RES0	Reserved for future type and index fields that can support nestedp2m and altp2m.
root_mfn	Root MFN of this domain's P2M
max_mapped_gfn	Maximum GFN mapped in this P2M
pginfos[N]	Array describing all pages in the P2M. Each entry has the same format as the page[] array in LU_PAGE_INFOS.

6.9 HAP_INFO

Describes additional information of the P2M if the domain is using HAP. Note that the pages are already described in the P2M_INFO record.



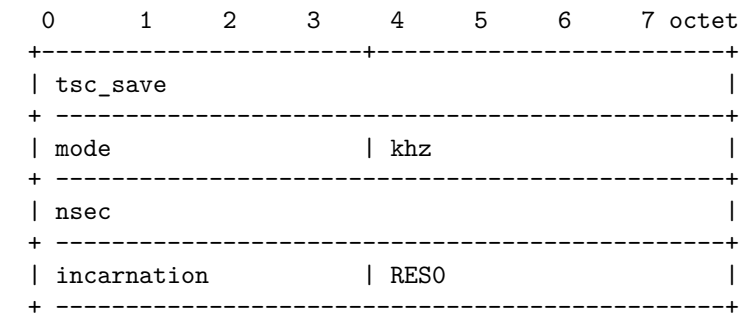
Field	Description
total_pages	Total number of pages that HAP can use

6.10 LU_X86_E820

This record contains the e820 map for the domain. The record is simply saved as a byte stream using the same format presented by a BIOS.

6.11 LU_X86_TSC_INFO

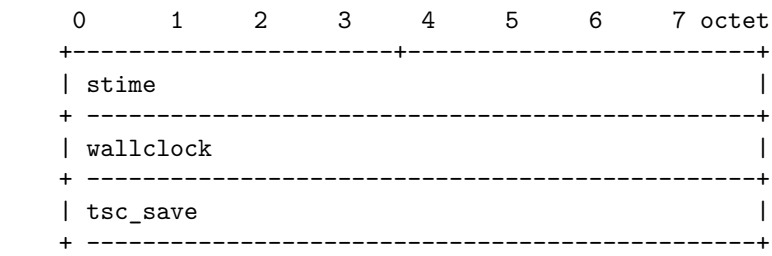
This record describes the TSC mode of a domain. It decides whether the TSC is native or emulated. This record must be restored before restoring the HVM TSC scaling and offset.



Field	Description
tsc_save	The TSC timestamp when this record is saved, to account for LU downtime.
mode	TSC mode, which decides whether the TSC is native or emulated. 0: native or emulated based on hardware 1: force emulated 2: force native 3: native but Xen manages TSC_AUX for RDTSCP
khz	TSC frequency, applicable for emulated.
nsec	TSC elapsed time in ns, applicable for emulated.
incarnation	How many times has this TSC been migrated.

6.12 CLOCK

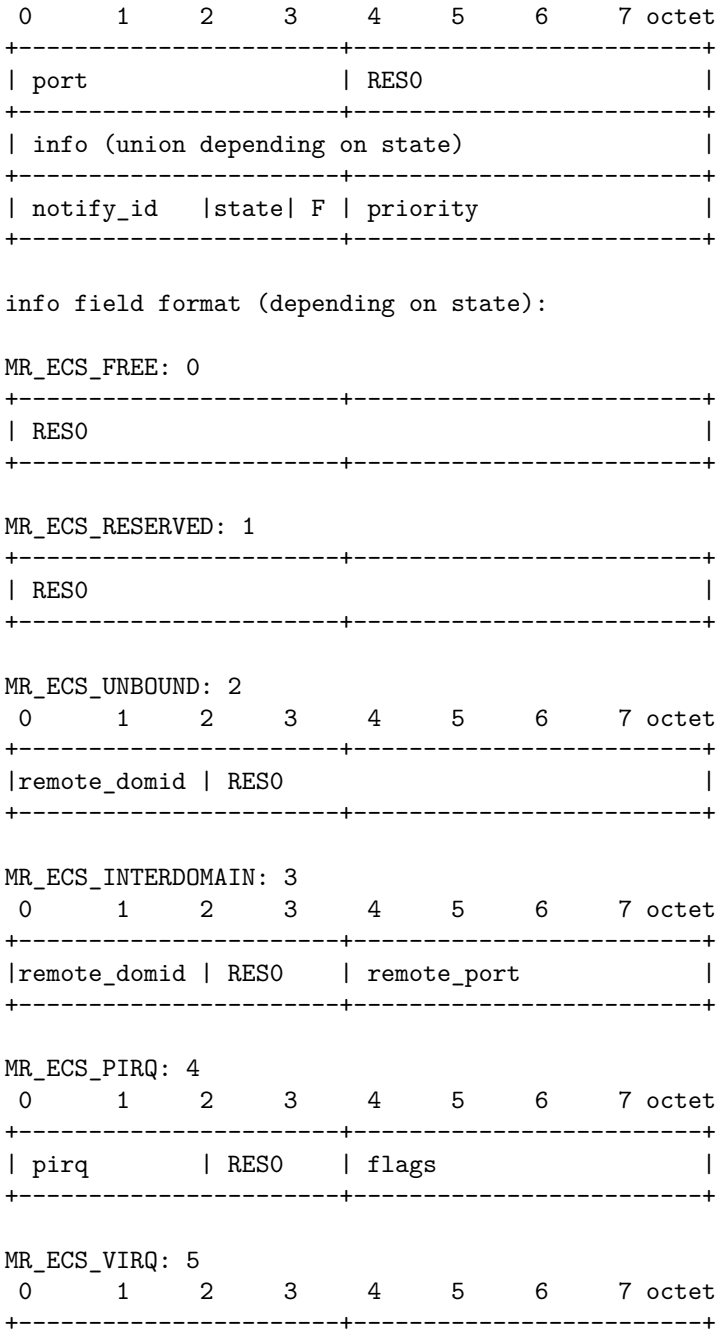
This record describes the time from the guest's PoV. All other time-related records (PV timers, HVM TSC, HVM HPET, etc.) must be restored after this record.



Field	Description
stime	Elapsed system time from the guest's PoV (in ns).
wallclock	Wallclock of this domain from the guest's PoV (in ns).
tsc_save	The TSC timestamp when this record is saved, to account for LU downtime.

6.13 EVTCHNS

This record describes each individual event channel state for a domain.



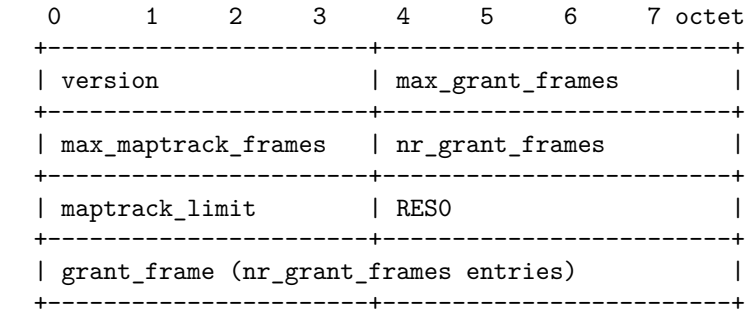
```
| virq        | RES0                                |
+-----------------------+-------------------------+

MR_ECS_IPI: 6
+-----------------------+-------------------------+
| RES0                                            |
+-----------------------+-------------------------+
```

Field	Description
port	Port number of this event channel.
info	Channel information depending on state. MR_ECS_UNBOUND remote_domid: the ID of the remote domain. MR_ECS_INTERDOMAIN remote_domid: the ID of the remote domain. remote_port: the port number of remote domain. MR_ECS_PIRQ pirq: pirq number. flags: LU_BIND_PIRQ___WILL_SHARE 1 whether this pirq is shareable MR_ECS_VIRQ virq: virq number.
notify_id	The vCPU to be notified for this event channel.
state	MR_ECS_* listed above.
F	Extra flags for the channel. bit 0: whether it is pending before FIFO array page is added. bit 1 - 7: reserved.
priority	The priority of this event channel for FIFO. Zeroed for 2L.

6.14 GRANT_TABLE

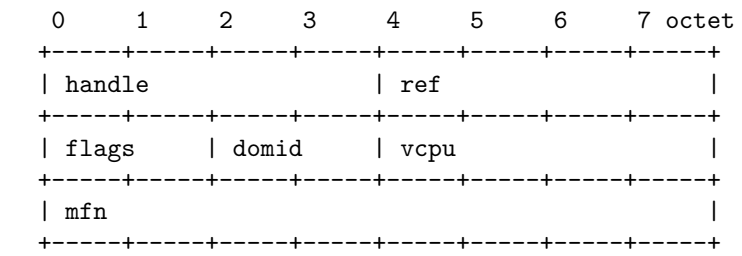
Describes the grant table state of each domain.



Field	Description
version	Grant table version number.
max_grant_frames	Maximum grant table frames for this domain.
max_maptrack_frames	Maximum maptrack frames for this domain.
nr_grant_frames	Number of frames shared with guest.
maptrack_limit	Number of available maptrack entries.
grant_frame	An array of MFNs of the grant table frames.

6.15 GRANT_MAPPINGS

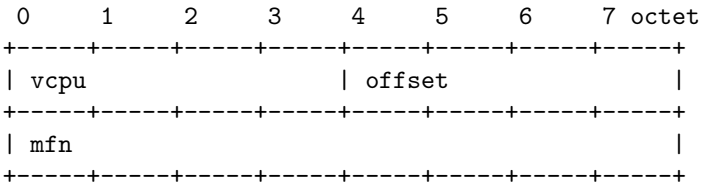
This record is an array containing an entry for each grant mapped in a domain. Each element of the array has the following format.



Field	Description
handle	tracking number of the mapping exposed to the domain
ref	grant reference number
flags	Flags identical to GNTMAP_* used by GNTTABOP_map_grant_ref
domid	Domain ID of the granter
vcpu	vCPU ID that issued the mapping request
mfn	MFN of the grant page

6.16 EVTCHN_FIFO_CONTROL_BLOCK

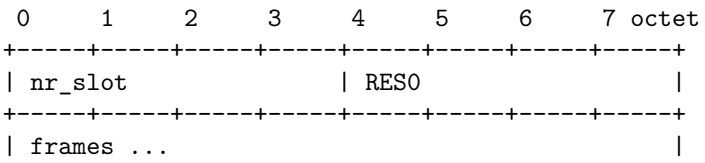
This record contains the per-vCPU FIFO event channel control block information.



Field	Description
vcpu	vCPU ID is bound to this event channel.
offset	Offset from the start of the page to the beginning of the control block.
mfn	MFN of the the page of the control block.

6.17 EVTCHN_FIFO_ARRAY

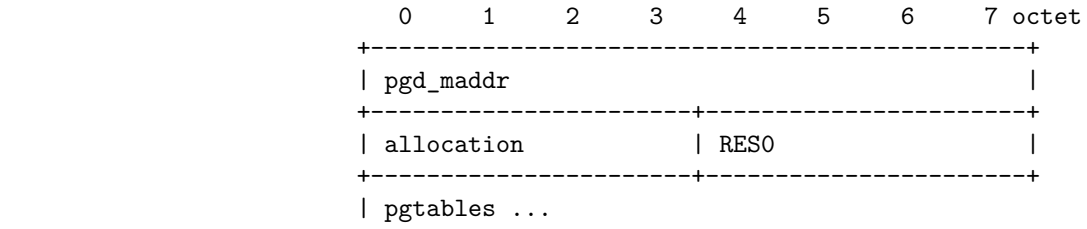
This record contains per-domain array of MFNs, where each MFN is mapped to a single slot (per event channel) of the domain.



Field	Description
nr_slot	Number of slots of the domain.
frames	An array of MFNs, each mapped to a single slot of the domain.

6.18 DOM_IOMMU_INFO

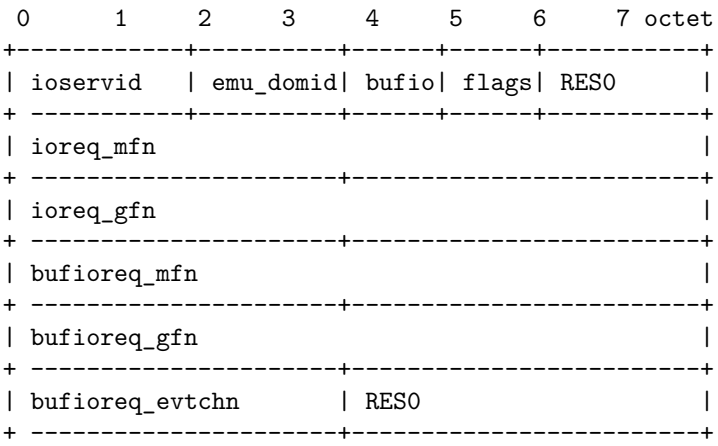
Per-domain IOMMU state.



Field	Description
pgd_maddr	The machine address of the root of the domain's IOMMU page tables.
allocation	The maximum number of IOMMU page tables the domain is allowed to allocate.
pgtables	The array of page table MFNs

6.19 IOSERV_INFO

This record describes an IOREQ server registered with the domain.



Field	Description
ioservid	IOREQ server ID, either the one returned by XEN_DMOP_create_ioreq_server, or the one reserved by Xen for the default IOREQ server.
emu_domid	ID of the domain where the emulator served by this IOREQ server runs.
bufio	One of HVM_IOREQSRV_BUFIOREQ_* indicating how this IOREQ server handles buffered I/O requests.
flags	Various attributes of this IOREQ server. Bit 0: Whether this IOREQ server is enabled. Bit 1: Whether this is the default IOREQ server for the domain. Bit 2-31: Reserved.
ioreq_mfn	MFN of the shared IOREQ page used by this IOREQ server for synchronous I/O requests. Can be INVALID_MFN if the page has not been allocated.
ioreq_gfn	GFN of the shared IOREQ page used by this IOREQ server for synchronous I/O requests. Can be INVALID_GFN if the page has not been allocated or is not mapped in the target domain's P2M.
bufioreq_mfn	MFN of the shared IOREQ page used by this IOREQ server for buffered I/O requests. Can be INVALID_MFN if the page has not been allocated. Undefined if the IOREQ server does not handle buffered I/O requests.

Field	Description
bufioreq_gfn	GFN of the shared IOREQ page used by this IOREQ server for buffered I/O requests. Can be INVALID_GFN if the page has not been allocated or is not mapped in the target domain's P2M. Undefined if the IOREQ server does not handle buffered I/O requests.
bufioreq_evtchn	Xen-side port number of the event channel used by this IOREQ server for buffered I/O requests. Undefined if the IOREQ server does not handle buffered I/O requests.

6.20 IOSERV_RANGES

This record complements the `IOSERV_INFO` record with information about the I/O ranges the IOREQ server has registered with Xen. Each `IOSERV_INFO` record may be followed by zero or more `IOSERV_RANGES` records, each describing one type of I/O range.

```
 0      1     2     3     4     5     6     7 octet
+-----------+---------+------------------------+
| ioservid  | RES0    | type                   |
+ ----------+-----------------------------------+
| ranges[0]                                    |
...
+----------------------+------------------------+
...
+----------------------+------------------------+
| ranges[N-1]                                  |
...
+----------------------+------------------------+
```

Field	Description
ioservid	ID of the IOREQ server which has registered these I/O ranges with Xen.
type	One of XEN_DMOP_IO_RANGE_* indicating the type of the I/O ranges in the ranges array.
ranges[N]	An array of I/O range entries (see layout below).

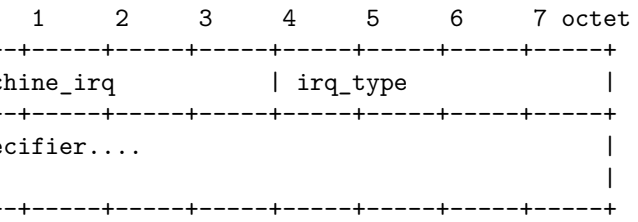
Each entry in `ranges[N]` has the following layout:

```
 0      1     2     3     4     5     6     7 octet
+----------------------+------------------------+
| start                                        |
+----------------------+------------------------+
| end                                          |
+----------------------+------------------------+
```

Field	Description
start	Start address of this I/O range. (The meaning of "address" depends on the I/O range type.)
end	End address of this I/O range.

6.21 REC_TYPE_X86_HVM_PT_PIRQS

Contains the list of physical interrupts map to an HVM domain. Each entry has the following layout:



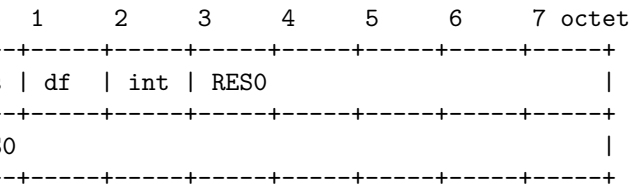
Field Description

machine_irq PIRQ number.

type Type of mapping 0: Legacy PCI interrupt 1: MSI

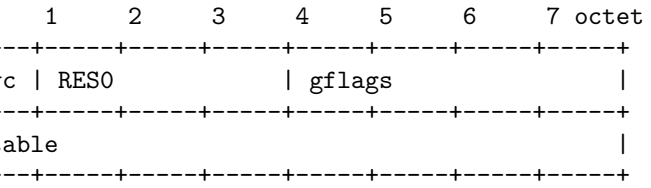
6.22 specifier Information specific to type

The specifier for type 0 (Legacy PCI Interrupt) is:



Field	Description
bus	Device 'bus'
df	Device 'devfn'
int	Interrupt pin

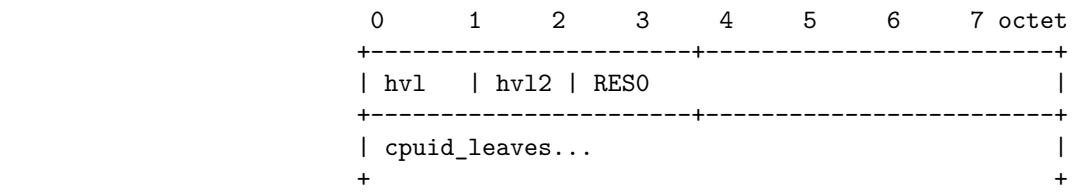
The specifier for type 1 (MSI) is:



Field	Description
gvc	Guest interrupt vector
gflags	Guest related flags Bit 0-7 : 0xff for multi-destination otherwise the destination ID Bit 8 : Redirection BIt 9 : Destination mode Bit 10-15: RES0 Bit 16-18: Delivery mode Bit 19 : Triggerring mode Bit 20-31: RES0
gtable	Guest Physical base address of the MSI-X table

REC_TYPE_CPUID_INF

Per-domain CPUID policy information, transferred as a list of <leaf, subleaf, eax, ebx, ecx, edx> entries.

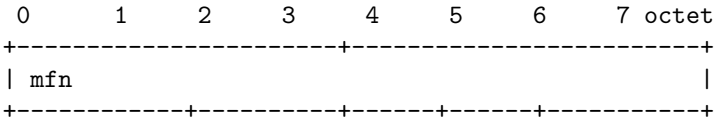


Field	Description
hvl	Toolstack selected max leaf (for non-Viridian domains)
hvl2	Toolstack selected max leaf (for Viridian domains)
cpuid_leaves	A list of entries for each <leaf, subleaf> pair. Each entry stores the value of the architectural register values <eax, ebc, ecx, edx> for the corresponding <leaf, subleaf> pair.

6.23 VLAPIC_MAPPING

This record describes a dummy page allocated by Xen for some x86 HVM domains to accelerate vLAPIC emulation using hardware-assisted virtualization. E.g. on Intel VMX, this page is known as the VMX APIC-access page, and Xen sets up MMIO redirection for vLAPIC accesses by mapping the base address of each vLAPIC to the said page in the domain's P2M (hence the name "vLAPIC mapping"). Note that the dummy page is only used for redirection and contains no useful data; it is different from the page that holds the vLAPIC register state (called "virtual- APIC page" by VMX), which is saved separately. After Live Update, the next Xen simply reuses the same vLAPIC mapping, instead of allocating a new dummy page.

If the domain does not have a vLAPIC mapping, this record is omitted.

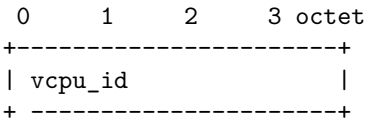


Field	Description
mfn	MFN of the domain's vLAPIC mapping, e.g. MFN of the APIC-access page on VMX.

7 Per-vCPU Records

7.1 VCPU Header

This is not a record, but a common header shared by all per-vCPU records that are introduced in this section. It appears at the beginning of each per-vCPU record, and identifies the vCPU the record is about.



Field	Description
vcpu_id	VCPU ID of the vCPU being reconstructed.

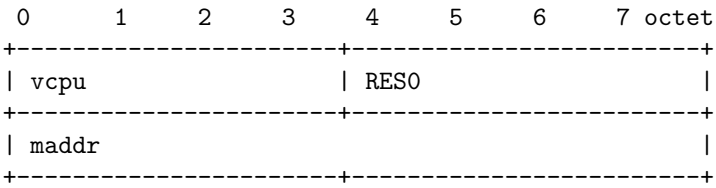
7.2 VCPU_INFO

This record describes a `vcpu_info` structure that a domain (either PV or HVM) has asked Xen to map into its memory via `VCPUOP_register_vcpu_info`. After LU, the next Xen needs to make sure the domain sees the same mapping.

Note that this record only applies to `vcpu_info`s that the domain explicitly maps or remaps via the said hypercall. For some vCPUs, the domain can also access `vcpu_info` via the default mappings, which are fixed offsets into the `shared_info` page. Those `vcpu_info`s should not be saved using this record.

While the domain uses a guest physical address to request the mapping of `vcpu_info`, the mapping may have disappeared (e.g. due to ballooning) by the time LU happens. For that reason, the mapping is transferred by its machine physical address instead of guest physical address, so as to simplify the restore side of things.

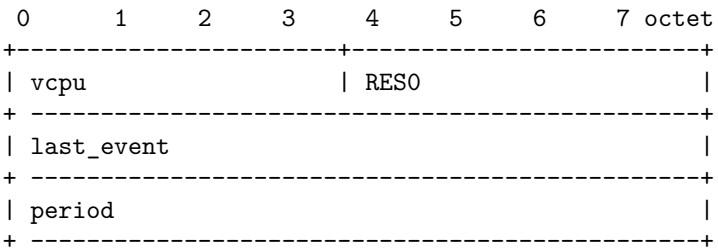
The size of `vcpu_info` is excluded from this record, because it is part of the Xen ABI. There is also guarantee that the structure does not cross a page boundary.



Field	Description
vcpu	Encapsulates the common vCPU header and identifies the vCPU for which the vcpu_info structure is created.
maddr	Machine physical address of the vcpu_info structure.

7.3 VCPU_TIMER_PERIODIC

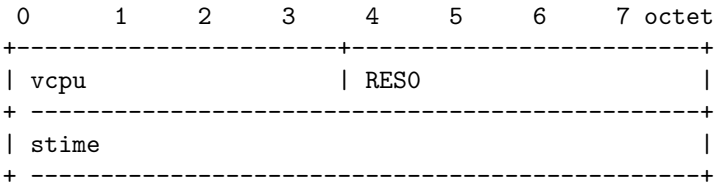
This record describes periodic timer of a vCPU.



Field	Description
vcpu	Per-vCPU record header.
last_event	The stime of the last event (in guest's PoV) fired.
period	The period of this timer, same unit as stime (ns).

7.4 VCPU_TIMER_SINGLESHOT

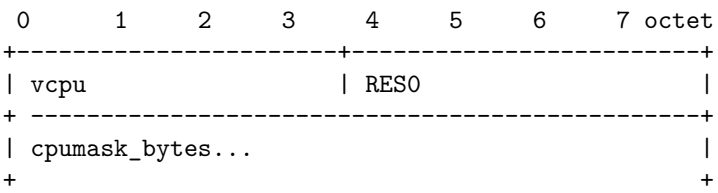
This record describes the singleshot timer of a vCPU.



Field	Description
vcpu	Per-vCPU record header.
stime	The stime (from the guest's PoV) at which this timer will fire.

7.5 VCPU_AFFINITY

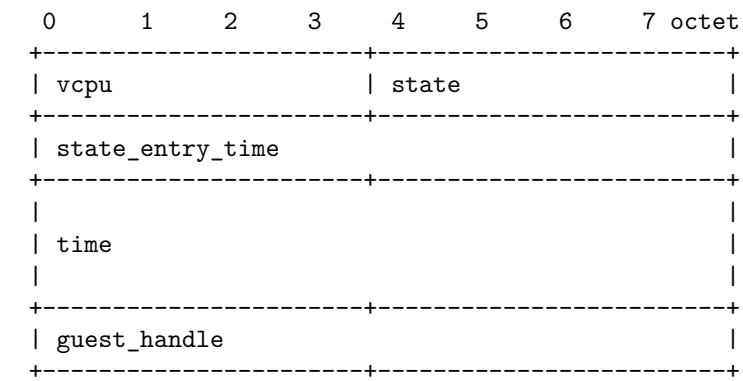
Information and bit vectors of vCPU pinning. This record depends on nr_present_cpus being the same after LU.



Field	Description
vcpu	Per-vCPU record header.
cpumask_bytes	The actual bit vector of hard and soft affinity. The vector is 2 * roundup(nr_present_cpus/BITS_PER_BYTE) bytes long. The first half is for the hard affinity, the rest being soft affinity.

7.6 VCPU_RUNSTATE

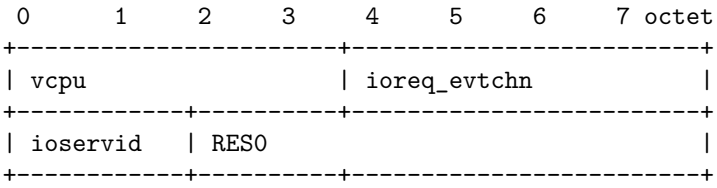
This record contains a snapshot of a vCPU's runstate data (see `struct vcpu_runstate_info`), as well as information about if/how that data is shared with the vCPU's domain via `VCPUOP_register_runstate_memory_area`. After LU, the next Xen needs to make sure that the domain sees consistent vCPU runstate data, and that it is still able to receive updates to the data in the same buffer that it registered with the previous Xen.



Field	Description
vcpu	Encapsulates the common vCPU header and identifies the vCPU this record is about.
state	VCPU's current state (`RUNSTATE_*`). See also `struct vcpu_runstate_info::state`.
state_entry_time	System time (ns) when the current state was entered. See also `struct vcpu_runstate_info::state_entry_time`.
time	Time (ns) spent in each `RUNSTATE_*`. See also `struct vcpu_runstate_info::time`.
guest_handle	Guest handle of a buffer for receiving the latest runstate data for this vCPU. Can be a null handle if the domain has not registered one via `VCPUOP_register_runstate_memory_area`. See also `struct vcpu::runstate_guest`.

7.7 IOSERV_VCPU

This record complements the `IOSERV_INFO` record with per-vCPU IOREQ server state. Each `IOSERV_INFO` record should be followed by `max_vcpus` (as in `LU_DOMAIN_INFO`) `IOSERV_VCPU` records, one for each vCPU in the domain that the IOREQ server targets.



Field	Description
vcpu	Per-vCPU record header.
ioreq_evtchn	Xen-side port number of the event channel used by this IOREQ server for synchronous I/O requests for this vCPU.
ioservid	ID of the IOREQ server this record is about.

7.8 HVM_VPMU_CONTEXT

This record contains the per-vCPU vPMU context (i.e. state of PMU registers) for HVM domains. It is omitted if the vPMU is not initialized for the vCPU, e.g. because PMU virtualization is globally disabled, or the guest has not used the vPMU.

```
 0      1     2     3     4     5     6     7 octet
+-----------------------+-------------------------+
| vcpu                  | RES0                    |
+-----------------------+-------------------------+
| blob ...
```

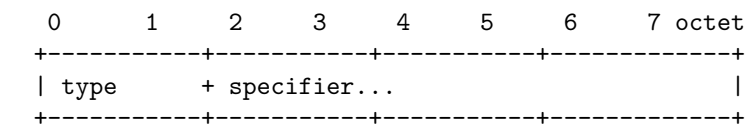
Field	Description
vcpu	Encapsulates the common vCPU header and identifies the vCPU this record is about.
blob	This vCPU's vPMU context blob. Its layout is vendor-specific, as defined in `struct xen_pmu_{intel,amd}_ctxt`, excluding metadata (non-register) fields.

8 Misc Records

8.1 LU_TIMESTAMP

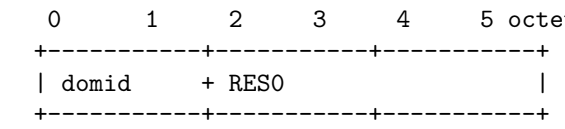
The record can be placed anywhere in the stream to collect a timestamp at pre-defined point in the save code. The restore code should not rely on all the type to be implemented.

Note that The timestamp is part of the per-record header.



Field	Description
type	Type of the timestamp: * 0: Live Update requested * 1: Domain quiesced * 2: All domains have been quiesced * 3: Start of saving the domains * 3: End of saving the domains * 4: End of saving a domain. A user can rely on LU_DOMAIN_INFO for the start * 5 - 65535: Reserved
specifier	Information specific to type (RES0 unless documented otherwise)

The following type contains more information in the specifier: * 1 (Domain quiesced)



Field	Description

8.2 domid ID of the domain quiesced

9 Future Extensions

All changes to this specification should bump the revision number in the title block.

The format may be extended by adding additional record types.

If there is no padding/reserved fields in an existing record, the extension must be done by adding a new record type. This allows old images with the old record

to still be restored.