Domain Save Image Format

David Vrabel <david.vrabel@citrix.com>

Draft D

Contents

1 Introduction

1.1 Revision History

Version	Date	Changes
Draft A	6 Feb 2014	Initial draft.
Draft B	10 Feb 2014	Corrected image header field widths.
		Minor updates and clarifications.
Draft C	27 Feb 2014	List feature excluded from this draft.
		Clarify which image versions a restore must support.
		x86 and ARM are always little-endian.
		Domain header: combine arch and type fields and add Xen major and minor version.
		Move checksum to end of record and include the header.
		Remove P2M record.
		Add some reserved bits to pfn fields in the PAGE_DATA record to allow for future expansion.
		List page types and note that XTAB can be used for unmapped pages at the end of a live migrations.
		Rename VCPU_INFO record to VCPU_COUNT.
		Add VCPU_CONTEXT_X1, and VCPU_CONTEXT_X2 records for the various extended VCPU context.
		Add array of P2M frame PFNs to X86_PV_INFO record.

Version	Date	Changes
Draft D	27 Feb 2014	Remove record checksum and option fields.
		Fail restores if there are unrecognized page types in a PAGE_DATA record.

1.2 Purpose

The *domain save image* is the context of a running domain used for snapshots of a domain or for transferring domains between hosts during migration.

There are a number of problems with the format of the domain save image used in Xen 4.4 and earlier (the *legacy format*).

- Dependant on toolstack word size. A number of fields within the image are native types such as `unsigned long` which have different sizes between 32-bit and 64-bit toolstacks. This prevents domains from being migrated between hosts running 32-bit and 64-bit toolstacks.
- There is no header identifying the image.
- The image has no version information.

A new format that addresses the above is required.

ARM does not yet have have a domain save image format specified and the format described in this specification should be suitable.

1.3 Not Yet Included

The following features are not yet fully specified and will be included in a future draft.

- HVM guests
- Remus
- Page data compression.
- ARM

2 Overview

The image format consists of two main sections:

- *Headers*
- *Records*

2.1 Headers

There are two headers: the *image header*, and the *domain header*. The image header describes the format of the image (version etc.). The *domain header* contains general information about the domain (architecture, type etc.).

2.2 Records

The main part of the format is a sequence of different *records*. Each record type contains information about the domain context. At a minimum there is a END record marking the end of the records section.

2.3 Fields

All the fields within the headers and records have a fixed width.

Fields are always aligned to their size.

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

Integer (numeric) fields in the image header are always in big-endian byte order.

Integer fields in the domain header and in the records are in the endianess described in the image header (which will typically be the native ordering).

3 Headers

3.1 Image Header

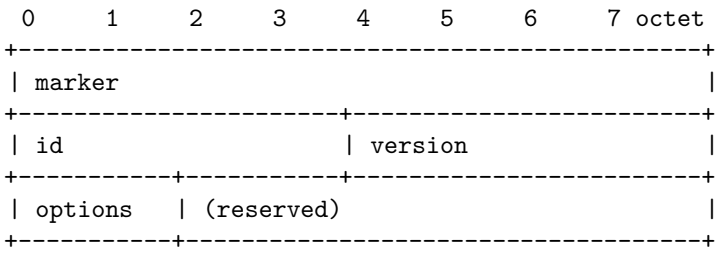
The image header identifies an image as a Xen domain save image. It includes the version of this specification that the image complies with.

Tools supporting version V of the specification shall always save images using version V. Tools shall support restoring from version V. If the previous Xen

release produced version V - 1 images, tools shall supported restoring from these. Tools may additionally support restoring from earlier versions.

The marker field can be used to distinguish between legacy images and those corresponding to this specification. Legacy images will have at one or more zero bits within the first 8 octets of the image.

Fields within the image header are always in *big-endian* byte order, regardless of the setting of the endianness bit.

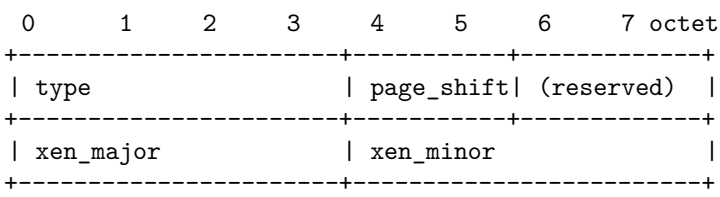


Field	Description
marker	0xFFFFFFFFFFFFFFFF.
id	0x58454E46 (“XENF” in ASCII).
version	0x00000001. The version of this specification.
options	bit 0: Endianness. 0 = little-endian, 1 = big-endian.
	bit 1-15: Reserved.

The endianness shall be 0 (little-endian) for images generated on an i386, x86_64, or arm host.

3.2 Domain Header

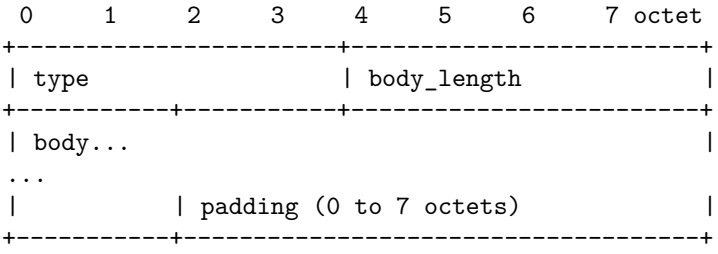
The domain header includes general properties of the domain.



Field	Description
type	0x0000: Reserved.
	0x0001: x86 PV.
	0x0002: x86 HVM.
	0x0003: x86 PVH.
	0x0004: ARM.
	0x0005 - 0xFFFFFFFF: Reserved.
page_shift	Size of a guest page as a power of two.
	i.e., page size = $2^{\text{page\_shift}}$.
xen_major	The Xen major version when this image was saved.
xen_minor	The Xen minor version when this image was saved.

4 Records

A record has a record header, type specific data and a trailing footer. If `body_length` is not a multiple of 8, the body is padded with zeroes to align the end of the record on an 8 octet boundary.



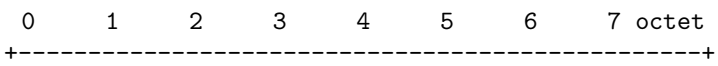
Field	Description
type	0x00000000: END
	0x00000001: PAGE_DATA
	0x00000002: VCPU_COUNT
	0x00000003: VCPU_CONTEXT
	0x00000004: VCPU_CONTEXT_X1
	0x00000005: VCPU_CONTEXT_X2
	0x00000006: X86_PV_INFO
	0x00000007 - 0x7FFFFFFF: Reserved for future *mandatory* records.
	0x80000000 - 0xFFFFFFFF: Reserved for future *optional* records.
body_length	Length in octets of the record body.
body	Record body of length `body_length` octets.
padding	0 to 7 octets of zeros to pad the whole record to a multiple of 8 octets.

Records may be *mandatory* or *optional*. Optional records have bit 31 set in their type. Restoring an image that has unrecognized or unsupported mandatory record must fail. The contents of optional records may be ignored during a restore.

The following sub-sections specify the record body format for each of the record types.

4.1 END

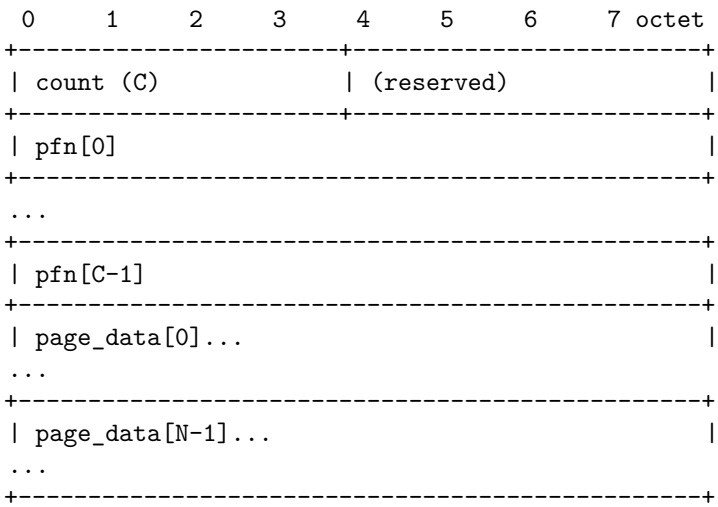
A end record marks the end of the image.



The end record contains no fields; its body_length is 0.

4.2 PAGE_DATA

The bulk of an image consists of many PAGE_DATA records containing the memory contents.



Field	Description
count	Number of pages described in this record.
pfn	An array of count PFNs and their types.
	Bit 63-60: XEN_DOMCTL_PFINFO_* type.
	Bit 59-52: Reserved.
	Bit 51-0: PFN.
page_data	page_size octets of uncompressed page contents for each page set as present in the pfn array.

PFINFO type	Value	Description
NOTAB	0x0	Normal page.
L1TAB	0x1	L1 page table page.
L2TAB	0x2	L2 page table page.
L3TAB	0x3	L3 page table page.
L4TAB	0x4	L4 page table page.
	0x5-0x8	Reserved.
L1TAB_PIN	0x9	L1 page table page (pinned).
L2TAB_PIN	0xA	L2 page table page (pinned).
L3TAB_PIN	0xB	L3 page table page (pinned).
L4TAB_PIN	0xC	L4 page table page (pinned).
BROKEN	0xD	Broken page.
XALLOC	0xE	Allocate only.
XTAB	0xF	Invalid page.

Table 6: XEN_DOMCTL_PFINFO_* Page Types.

PFNs with type `BROKEN`, `XALLOC`, or `XTAB` do not have any corresponding `page_data`.

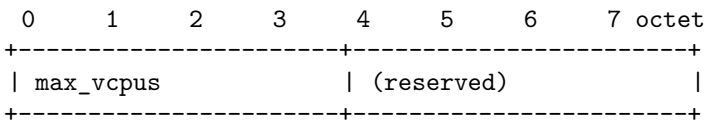
The saver uses the `XTAB` type for PFNs that become invalid in the guest's P2M table during a live migration[1].

Restoring an image with unrecognized page types shall fail.

[1] In the legacy format, this is the list of unmapped PFNs in the tail.

4.3 VCPU_COUNT

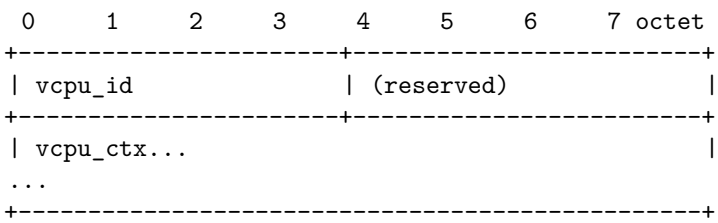
The VCPU_COUNT record includes the maximum number of VCPUs. This will be followed a VCPU_CONTEXT record for each online VCPU.



Field	Description
max_vcpus	Maximum number of VCPUs.

4.4 VCPU_CONTEXT

The context for a single VCPU, as accesses by the XEN_DOMCTL_getvcpucontext and XEN_DOMCTL_setvcpucontext hypercall sub-ops.



Field	Description
vcpu_id	The VCPU ID.
vcpu_ctx	Context for this VCPU.

4.5 VCPU_CONTEXT_X1

Additional context for a single VCPU, as accessed by the XEN_DOMCTL_get_ext_vcpucontext and XEN_DOMCTL_set_ext_vcpucontext hypercall sub-ops.

```
 0     1     2     3     4     5     6     7 octet
+-----------------------+-------------------------+
| vcpu_id               | (reserved)              |
+-----------------------+-------------------------+
| vcpu_ctx...                                     |
...
+-------------------------------------------------+
```

Field	Description
vcpu_id	The VCPU ID.
vcpu_ctx	Context for this VCPU.

4.6 VCPU_CONTEXT_X2

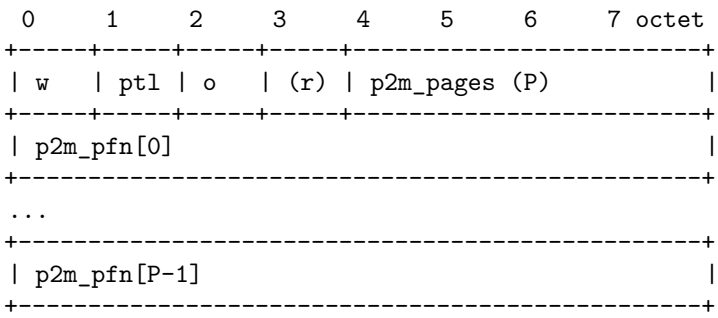
Additional context for a single VCPU, as accessed by the XEN_DOMCTL_getvcpuextstate and XEN_DOMCTL_setvcpuextstate hypercall sub-ops.

```
 0     1     2     3     4     5     6     7 octet
+-----------------------+-------------------------+
| vcpu_id               | (reserved)              |
+-----------------------+-------------------------+
| vcpu_ctx...                                     |
...
+-------------------------------------------------+
```

Field	Description
vcpu_id	The VCPU ID.
vcpu_ctx	Context for this VCPU.

4.7 X86_PV_INFO

[This record replaces part of the extended-info chunk, and the p2m frame list]



Field	Description
guest_width (w)	Guest width in octets (either 4 or 8).
pt_levels (ptl)	Number of page table levels (either 3 or 4).
options (o)	Bit 0: 0 - no VMASST_pae_extended_cr3, 1 - VMASST_pae_extended_cr3. Bit 1-7: Reserved.
p2m_pages	Guest's P2M table size in pages.
p2m_pfn	Array of PFNs containing the guest's P2M table.

5 Layout

The set of valid records depends on the guest architecture and type.

5.1 x86 PV Guest

An x86 PV guest image will have in this order:

1. Image header
2. Domain header
3. X86_PV_INFO record
4. Many PAGE_DATA records
5. VCPU_COUNT record
6. VCPU context records for each online VCPU
    a. VCPU_CONTEXT record
    b. VCPU_CONTEXT_X1 record
    c. VCPU_CONTEXT_X2 record
7. END record

6 Legacy Images (x86 only)

Restoring legacy images from older tools shall be handled by translating the legacy format image into this new format.

It shall not be possible to save in the legacy format.

There are two different legacy images depending on whether they were generated by a 32-bit or a 64-bit toolstack. These shall be distinguished by inspecting octets 4-7 in the image. If these are zero then it is a 64-bit image.

Toolstack	Field	Value
64-bit	Bit 31-63 of the p2m_size field	0 (since p2m_size $< 2^{32}$)
32-bit	extended-info chunk ID (PV)	0xFFFFFFFF
32-bit	Chunk type (HVM)	< 0
32-bit	Page count (HVM)	> 0

Table 12: Possible values for octet 4-7 in legacy images

This assumes the presence of the extended-info chunk which was introduced in Xen 3.0.

7 Future Extensions

All changes to the image or domain headers require the image version to be increased.

The format may be extended by adding additional record types.

Extending an existing record type must be done by adding a new record type. This allows old images with the old record to still be restored.

The image header may only be extended by *appending* additional fields. In particular, the `marker`, `id` and `version` fields must never change size or location.