Customizing VIBs with VIB Author

ESXi 5.0

The VIB Author tool allows ESXi administrator to create custom VIBs at the CommunitySupported level. The custom VIBs can then be used in house, or made available to a wider audience.

IMPORTANT If you add a CommunitySupported VIB to an ESXi host, you must first change the host's acceptance level to CommunitySupported. If you encounter problems with an ESXi host that is at the CommunitySupported acceptance level, VMware Support might ask you to remove the custom VIB, as outlined in the support policies:

"*Where VMware suspects that a problem may be related to modified code, VMware in its sole discretion, reserves the right to request that the modified code be removed. If VMware cannot directly identify the root cause of a problem, or reasonably suspects that the problem is related to modified code, you shall be informed that you may obtain additional assistance directly from various product discussion forums or via engagement for an additional fee with VMware's consulting services group.*"

The VIB Author tool allows you to create VIBs using custom VIB descriptor files and custom payload files. The tool also allows you to sign and verify a VIB, extract and display VIB information, and publish the VIB as an offline depot ZIP file. This document explains how to use the tool and includes reference information.

Getting Started

This document assumes that you have a working familiarity with the following:

- Linux Operating System—General working knowledge.
- VMware products—A general understanding of ESXi and vSphere.

To get started, download and install the VIB Author RPMs from the VMware Labs Web site. VIB Author is supported on only Linux.

Then follow the instructions in this document to create and stage your VIB.

The VIB Author Tool includes man pages for the VIB Author tool. From the `/opt/vmware/vibtools/doc/man/man9` directory, run `man ./vibauthor.9`

To learn about building and packaging image profiles, see the *vSphere Installation and Setup* documentation, available at http://pubs.vmware.com.

Terms

This document uses the following technical terms.

acceptance level	Acceptance levels are categories used to summarize the level of testing and verification a VIB has undergone before its release and, consequently, the level of support which will be extended to customers who use it. See "Acceptance Levels" on page 9 for more information.
image profile	An image profile defines an ESXi image and consists of VIBs (software packages). An image profile always includes a base VIB, and might include additional VIBs. You examine and define an image profile using the Image Builder PowerCLI. See the Image Builder sections in the *Installation and Setup* document, available at http://www.pubs.vmware.com, for detailed information on using Image Builder to create image profiles.
offline bundle	A ZIP file that contains VIBs. An offline bundle can be installed using ESXCLI or using vSphere Update Manager.
VIB	A VIB is an ESXi software package. VMware and its partners package solutions, drivers, CIM providers, and applications that extend the ESXi platform as VIBs. You can create your won custom VIB with VIB Author. You can use VIBs to create and customize image profiles or to upgrade ESXi hosts by installing VIBs asynchronously onto the hosts
VIB metadata	An XML file (`descriptor.xml`) that describes the contents of the VIB. A template for `descriptor-template.xml` is provided with the tool and filled out by the user.
VIB payload	The VIB payload contains the files that the VIB consists of. When a VIB is added to an ESXi image, the files in the VIB payload are installed on that host. When a VIB is removed from an ESXi image, the payload files are removed.

VIBs and VIB Formats

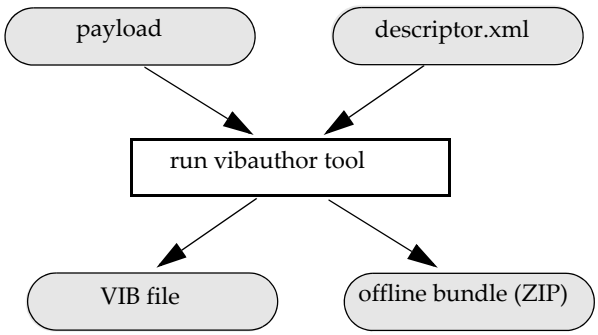
A VIB has the following components:

- A `descriptor.xml` file that contains the VIB metadata (see "Creating or Editing the VIB Descriptor XML" on page 3).
- sig.pkcs7 - A gzipped PKCS7 Signature. If the VIB is not signed, an empty sig.pkcs7 is required. The signature is not loaded by the bootloader.
- Directory that includes the file that will be included in the VIB (payload).

After a VIB is created it must be packaged into a format that can be consumed by ESXCLI or vSphere Update Manager before it is distributed. VIBs you create with the VIB Author tool are bootbank VIBs. The payload is installed in the bootbank partition on the ESXi host.

VIB Author Workflow

The VIB Author tool creates a VIB that contains the payload and the metadata file. The tool can also publish the VIB as an offline bundle. A simplified workflow is outlined below.



Creating a VIB

Creating a VIB consists of these tasks, performed in a terminal window:

- "Creating or Editing the VIB Descriptor XML" on page 3
- "Staging the Files" on page 5
- Creating a VIB, optionally signing the VIB, or creating the offline bundle. See "VIB Creation Scenarios" on page 6

NOTE Instead of creating a VIB from scratch, you can extract an existing VIB, modify one or more of the files, and create a VIB from that set of files. See "VIB Extraction and Modification Scenarios" on page 7.

Creating or Editing the VIB Descriptor XML

The VIB Author tool requires you to provide information about the files that will be included in the VIB. This information is stored in an XML file that is distributed with the VIB.

A sample descriptor file (`descriptor-template.xml`) that you can use as a starting point for creating this file is installed in the `/opt/vmware/vibtools/sample/` directory.

Understanding Dependencies, Conflicts, and Replacements

Modify the contents of the `<depends>`, `<replaces>`, `<conflicts>` and `<provides>` fields to correctly reflect the dependencies, obsolescense, and conflicts that apply to the VIB payload. These fields should contain the VIB name to identify the VIB.

- **Depends Field**. Use the `<depends>` field to indicate dependencies. Dependencies can be files or libraries that VIB payload requires. In the rare case that your software is dependent on a change that occurred in an ESXi update release, you can specify a dependency on the ESXi version, installer version, or ESXi image version.

  NOTE VIBs must not create a self-dependency, where the VIB both provides and consumes a particular dependency

- **Replaces Field.** Use the `<replaces>` field to replace an existing package with the one contained in the VIB, only when the VIB has changed names between versions. This field signifies that ESXi should replace a specific, older installed version of the VIB contents with the newer version that is contained in the VIB. This field allows the upgrade tools to detect a VIB that should be replaced when the name has changed. Do not use this field for a simple update where no name change has been made; use the version field.

If the VIB you are creating is an update to an existing VIB, and the name and contents of the VIB have not changed since the last release, use a newer version number to trigger an update. Do not use the `<replaces>` field for this purpose.

NOTE Only replace your own VIBs. Never use the `<replaces>` field to replace VIBs provided by VMware or to replace other third-party VIBs.

- **Conflicts Field.** The `<conflicts>` field is used to prevent installation of different packages that will not operate correctly if both are installed.
- **Provides Field.** Exposes functionality provided by the VIB in the form of an API string. So a VIB that `<depends>` on 'abc' would have the dependency satisfied by any other VIB that `<provides>` 'abc'. If your package is required by other VIBs, define a <provides> field. Use reverse DNS form in this field, for example, `com.vmware.dir.content`

VIB XML Metadata Fields

Create a copy of the `descriptor-template.xml` file, and modify the fields to reflect the settings for your package. The following table describes the fields in the VIB data file.

Table 1. VIB Metadata (All VIB Types)

Field	Input/ Generated	Description
type	I	Type of VIB.
name	I	VIB name, with a length less than or equal to 35 characters.
version	I	VIB version, with a length less than or equal to 35 characters. This must be specified in version-release format. See "Versioning VIBs" on page 7. Unlike versions in Depends, Conflicts, Replaces and Provides fields, the release version portion is mandatory.
vendor	I	Vendor publishing the VIB.
summary	I	Short summary of package functionality.
description	I	Full description of package functionality.
release-date	I/G	Date of VIB publication or release, as a floating point number.
urls	I	A list of URLs for KB articles or other more in depth relevant information. Each URL is accompanied by a short string key, 20 characters or less, giving a unique identifier.
depends	I/G	Other VIBs that must be installed along with this one. See "Understanding Dependencies, Conflicts, and Replacements" on page 3.
conflicts	I/G	VIBs that cannot be installed together with this one. See "Understanding Dependencies, Conflicts, and Replacements" on page 3.
replaces	I/G	Like obsoletes in RPM; VIB replaces other VIBs. See "Understanding Dependencies, Conflicts, and Replacements" on page 3.
provides	I	VIB provides virtual packages, for example, `DriverAPI-10.0`. See "Understanding Dependencies, Conflicts, and Replacements" on page 3.
tags	I	Unordered list of tags, for example, `driver` or `security-only`.
checksum	G	MD5/SHA checksum of the VIB package.
packed-size	G	Size of VIB package.
relative-path	G	Relative path of VIB package within depot.

Table 2. VIB Metadata (Bootbank VIBs)

Field	Input /Generated	Description
maintenance-mode	I	True if the ESXi host has to be in maintenance mode when installing this VIB.
hwplatform	I	This VIB targets specific (OEM) hardware.
file-list	G	List of files packaged with VIB.
acceptance-level	I	Specifies an acceptance level for the VIB, which will dictate the standards that will be used for validating the VIB signature and whether or not the VIB will be allowed to use restricted VIB features. Specify CommunitySupported. The tool allows you to generate a VIB with a different acceptance level, but you cannot install that VIB on your ESXi host.
live-installs-allowed	I	Can install and activate the VIB without a reboot.
live-removals-allowed	I	Can remove and deactivate the VIB without a reboot.
cimom-restart	I	True if cimom needs to be restarted after installation. Enable this field for CIM provider VIBs.
stateless-ready	I	True if the VIB supports host profiles and other technologies such that it can be successfully deployed on a stateless host.
resources	I	Specify CPU and memory resource limits.
resource-group-path	I	Specify the resource group path name, for example, `host/vim/vmvisor/hostd`.
overlay	I	Boolean, this VIB is an overlay VIB.
payloads	I/G	List of bootbank payloads in the VIB package. See "Staging the Files" on page 5.
install-date	G	Installation timestamp.

Staging the Files

Before you run the VIB Author tool, you must stage all files in the staging directory.

The VMware Labs version of VIB Author supports a new `compose` option. The old `create` option is deprecated in this version. Before you use the `compose` option, place both the descriptor file and payload files in a staging directory. Organize the staging directory as follows:

```
<stage_dir> / descriptor.xml
<stage_dir> / <payload_dir1> / ...
<stage_dir> / <payload_dir2> / …
```

Each payload directory contains the files within the payload. The structure within the directory is mirrored when the VIB payload files are installed on the ESXi host. The payload metadata (`<payload element>`) is defined in `descriptor.xml`. It contains both payload name and payload type. Name the directory by the payload name. The VIB Author tool compresses the payload into the specified type (VGZ or TGZ).

NOTE You must define the payload information, including payload name and payload type, in the metadata file. The VIB Author tool check the metadata file for the location of payload files to include in the VIB and for compression information for each file.

For example if you have a VGZ type payload with name `payload1`, set up the files as follows:

Directory for payload files: *stage_dir*`/payload1`

Payload definition in `descriptor.xml` metadata file:

```
<payloads>
   <payload name="payload1" type="vgz">
</payload> </payloads>
```

Use only explicitly allowed paths from the following list.

NOTE If you want to create a VIB with files that are in other locations, you must use the `--force` flag.

- `etc/vmware/shutdown/shutdown/`
- `etc/vmware/pciid/`
- `etc/vmware/vm-support/`
- `etc/vmware/firewall/`
- `etc/vmware/service/`
- `etc/cim/openwsman/`
- `opt/`
- `usr/lib/cim/`
- `usr/lib/pycim/`
- `usr/lib/hostprofiles/plugins/`
- `usr/lib/vmware/`
- `usr/lib/vmware-debug/`
- `var/lib/sfcb/registration/`
- `/VIB/`
- `etc/vmware/driver.map.d`
- `usr/share/hwdata/driver.pciids.d`

NOTE The VIB directory is a special directory for installation scripts. The only files that can be placed in this directory are installation scripts.

VMware suggests that you append the allowed path with a vendor-specific directory for files. For instance, for SomeCompany's CIM providers, you might use `/etc/cim/somecompany/.`

VIB Creation Scenarios

The following scenarios provide the basic steps involved in creating VIBs. Run the commands in a terminal window. You have the following choices.

- "Creating a VIB" on page 6. You can create a VIB with a globally unique name and then use ESXCLI to add the VIB to your hosts. You can optionally sign your VIB.
- "Creating an Offline Bundle" on page 7. You can create an offline bundle ZIP file for easy distribution and for use with vSphere Update Manager. You cannot sign an offline bundle.

Creating a VIB

To create a VIB, follow these steps.

1 Modify the VIB data template, as described in "Creating or Editing the VIB Descriptor XML" on page 3.

   Add all required and optional metadata for your VIB.

2 Copy all the files to a staging directory. See "Staging the Files" on page 5.

3 Run the VIB Author tool to create a VIB file.

   ```
   vibauthor -C -t stage-dir -v e1000.vib [options]
   ```

   NOTE The VIB file name must be globally unique.

Signing a VIB

The ESXi host does not check the signature of VIBs at the CommunitySupported level, so you do not have to sign your VIB before it can be installed on an ESXi host. However, you must lower the host acceptance level

to CommunitySupported before you can install your VIB on a host. You can lower the host acceptance level with ESXCLI commands. See the *vSphere Installation and Setup* documentation.

IMPORTANT VMware does not provide technical support for community VIBs. See https://www.vmware.com/support/policies. Look in particular at the section on modified code.

If you encounter problems with an ESXi host that is at the CommunitySupported acceptance level, VMware Support might ask you to remove the custom VIB.

You can optionally sign a CommunitySupported VIB with `vibauthor` yourself.

To sign a VIB, follow these steps.

1 Generate a private/public key pair and a certificate.

2 Invoke the VIB authoring tool to sign the VIB you created.

```
vibauthor –s –v e1000.vib –k private_keyfile –r certfile
```

NOTE The VIB file name must be globally unique.

You can also create a signed VIB in one command.

```
vibauthor –C –t stage–dir –s –k private_keyfile –r certfile –v e1000.vib
```

Creating an Offline Bundle

You can create an offline bundle at the same time that you are creating and signing the VIB. You can also create the offline bundle after you have created and optionally signed the VIB.

To create an offline bundle, run the following command.

```
vibauthor –C –t stage–dir –v e1000.vib –O e1000.zip
```

You can use the offline bundle with Image Builder and with vSphere Update Manager (VUM).

VIB Extraction and Modification Scenarios

You can extract a VIB by using the [`--expand|–e`] and [`––compose|–C`] command-line options. You can then modify files in the payload, and recreate the VIB with the VIB Author tool.

NOTE If you modify the payload and recreate the VIB, the existing signature becomes invalid. You can sign the VIB yourself to create a signed CommunitySupported VIB. See "Signing" on page 8.

- Extract a VIB from a VIB file

  ```
  vibauthor –e –v vib_file –o output_dir
  ```

- Extract a VIB from an offline bundle

  a Show the list of VIBs in the offline bundle.

  ```
  vibauthor –e –z offlinebundle.zip
  ```

  a Extract one of the VIBs from the offline bundle.

  ```
  vibauthor –e –z offlinebundle.zip –v VIB_ID –o output_dir
  ```

Versioning VIBs

Because VIBs are intended to allow the ESXi installer to handle updates as well as initial installs, it is important to use the versioning variables in the VIB correctly.

The package version can have a length of up to 35 characters. It includes the version number, immediately followed by a hyphen (-) followed by the release version, as in the following example:

```
5.0.0–1.0vmw
```

Both the version number and the release version number consist of one or more characters in the set [a-a-Z0-9], with periods separating two or more groups.

VIBs for Upgrades

You can perform upgrades where VIB names match or where VIB names do not match.

- Upgrade where the VIB names match

  If you upgrade the contents of a VIB, if possible ensure that the name of the new VIB is the same as that of the VIB it replaces. If the names are the same, the installer will check the version numbers in the XML files associated with each VIB and keep or install the VIB with the higher number. Keeping the same VIB name means that the version number is used to determine which VIB should be used.

- Upgrade where the VIB names do not match

  If the name of the new VIB cannot be the same as that of the old VIB, or if a single new VIB should replace multiple older VIBs, or use the <replaces> tag in the VIB `descriptor.xml` file to list all the old VIBs that should be replaced. For example, if the old VIB had the name extension-A but the new VIB has the name extension-B, add an entry like the following to your `descriptor.xml` file.

  ```
  <replaces>
         <constraint name="extension-A" />
  </replaces>
  ```

Distributing Your Software

You can make your VIB available to others in your companies or to the greater VMware Community. The following distributions methods are supported.

- **Single VIBs.** The simplest way to distribute your software is as a single VIB. You can install a single VIB using ESXCLI commands. vSphere Update Manger and Image Builder cannot use an individual VIB; they require an offline bundle.
- **Offline Bundles.** Offline bundles are self-contained files that can be consumed by tools and utilities from VMware including ESXCLI, vSphere Update Manager, and Image Builder. Offline bundles are not restricted in size, but it is a good practice to keep them small.

Signing

The VIB Author Tool allows you to create a CommunitySupported VIB. You can optionally sign the VIB.

There should be one signature for an entire VIB. The descriptor data contained in the `descriptor.xml` is signed using a detached PKCS7 signature. The PKCS7 signed VIB should use SHA-256 as the digest algorithm for verifying VIB integrity. The PKCS7 structure must include the signer's certificate. It may optionally include additional certificates needed to establish a chain of trust from the signer to VMware (or another trusted entity). The PKCS7 signature data is added to the VIB's `sig.pkcs7` component as PEM-encoded data.

Unsigned VIBs must contain an empty `sig.pkcs7` component.

A VIB signature is considered valid if the VIB is valid, and the signer can be trusted. A signer is considered trusted if:

- a chain of trust can be established between the signer's certificate and the certificate of a trusted authority. A certificate or list of certificates of trusted authorities should be an input to any function performing validation.
- neither the signer's certificate nor any certificate in the chain of certificates between the signer's certificate and the certificate of the trusted authority has been revoked. Revoked certificates are published in a Certificate Revocation List. Any function performing signature validation should accept and process a Certification Revocation List as input.

The message content is considered valid if the signed digest within the PKCS7 signature matches the digest calculated over the message (\descriptor.xml) contents.

Acceptance Levels

Acceptance levels are categories used to summarize the level of testing and verification a VIB has undergone before its release and, consequently, the level of support which will be extended to customers who use it. Acceptance levels of a host is based on the lowest acceptance level of any VIB on the host.

IMPORTANT If you add a CommunitySupported VIB to an ESXi host, you must first change the host's acceptance level. If you encounter problems with an ESXi host that is at the CommunitySupported acceptance level, VMware Support might ask you to remove the custom VIB.

VIBs can belong to one of the following acceptance levels:

VMwareCertified	The VMwareCertified acceptance level has the most stringent requirements. VIBs with this level go through thorough testing fully equivalent to VMware in-house Quality Assurance testing for the same technology. Today, only IOVP drivers are published at this level. VMware takes support calls for VIBs with this acceptance level.
VMwareAccepted	VIBs with this acceptance level go through verification testing, but the tests do not fully test every function of the software. The partner runs the tests and VMware verifies the result. Today, CIM providers and PSA plugins are among the VIBs published at this level. VMware directs support calls for VIBs with this acceptance level to the partner's support organization.
PartnerSupported	VIBs with the PartnerSupported acceptance level are published by a partner that VMware trusts. The partner performs all testing. VMware does not verify the results. This level is used for a new or nonmainstream technology that partners want to enable for VMware systems. Today, driver VIB technologies such as Infiniband, ATAoE, and SSD are at this level with nonstandard hardware drivers. VMware directs support calls for VIBs with this acceptance level to the partner's support organization.
CommunitySupported	The Community Supported acceptance level is for VIBs created by individuals or companies outside of VMware partner programs. VIBs at this level have not gone through any VMware-approved testing program.

Acceptance Levels and VIB Signing

VIBs with the VMwareCertified, VMwareAccepted and PartnerSupported levels must be signed with a private key, and the certificate corresponding to that private key must be included in the signature data. VIBs with acceptance levels that require signing must have a valid signature in addition to being signed by a trusted authority. Community supported VIBs do not need to be signed.

VMware itself must sign all VIBs which have an acceptance level of VMware Certified or VMware Accepted. VIBs with acceptance level Partner Signed are signed by external entities entrusted by VMware (Partners). All signed VIBs should have gone through an official VMware certification program.

VIB Author only generates Community Supported VIBs, VMware does not test or sign these VIBs.

VIB Author Command Line Interface Reference

The VIB Author tool includes a number of options.

NOTE The --create and --unpack options are deprecated. Use --compose and --expand instead.

```
vibauthor ((-C -t unpack-dir [-O file.zip]) |
    (-c -d descriptor.xml [-t stage-dir] [[-p file,type[,<name>[,boot]]] -p ...]))
    [-v file.vib]
    [-s -k private -r cert]
    [-f]
vibauthor -i -v file.vib [-x]
vibauthor ((-e ((-z depot.zip -v vib-id) | (-v file.vib))) | (-u -v file.vib)) [-o dir] [-f]
vibauthor -s -v file.vib -k private -r cert
```

```
vibauthor –a –v file.vib –r cert [–r ...]
```

The following list describes each option in detail and provides some shortcuts.

`––compose | –C`

Creates a VIB. Use the `––vib` option to give the resulting VIB a specific name or to write the VIB to a specific location. If not provided, the VIB name is based on the VIB ID.

`––create | –c`

This option is deprecated and will be removed from the next release. Use the `––compose` option instead.

`––descriptor=file | –d file`

This option is deprecated and will be removed from the next release.

`––stage–dir=dir | –t dir`

When used with the `––compose` option, specifies the staging directory, which contains both the descriptor and the extracted payloads.

`––payload=file,type[,name[,boot]] | –p file,type[,name[,boot]]`

This option is deprecated and will be removed from the next release.

`––output–dir=dir | –o dir`

Specifies the output location when one or more files need to be created as a result of the command (excludes temporary files that are cleaned up before the command completes). The default is the current directory, if unspecified.

When used with `––unpack`, `vibauthor` overwrites the directory with the raw contents of the VIB. When used with `––expand`, `vibauthor` overwrites the directory with the descriptor and the extracted payloads of the VIB.

`––depot=file | –z file`

Sets the path for the offline depot file when used with `––expand`.

NOTE The offline depot file path must be globally unique.

Use `––vib` to specify the ID of the VIB that you want to unpack. If you do not specify `––vib`, the command lists the VIB IDs in the offline depot.

`––vib=file | –v file`

Sets the name for the output VIB file when used with `––compose`. Sets the ID of the VIB when used with `––depot`.

NOTE The VIB file name must be globally unique.

If you do not include a `.vib` extension, it will be added to the specified file name. If the option is omitted then the file name defaults to the VIB unique identifier and the VIB is written to the default directory (current).

When used with `––sign`, the VIB is treated as input and output.

When used with `––unpack` or `––expand`, the VIB is input only.

`––sign | –s`

Causes a VIB to be signed using a private key specified using the `––key` and `––cert` options. This option can be used during VIB creation or at a later time.

If the VIB has already been signed, the new signature replaces the old only if the --force flag has also been specified. Otherwise the tool prints a message and exit.

`––authenticate | –a`

Checks the signatures on the VIB for authenticity. Must provide a CA certificate using `--cert`.

`--force | -f`

Overrides certain warnings. This carries a certain amount of risk, depending on the error being overridden, so exercise caution. In particular, this option allows you to do the following:

- Override a previously signed VIB or VIB payload file with a new signature or file.
- Overwrite files if they already exist.
- Use files that are not in the preset directory structure.

`--key=file | -k file`

Specifies a key file. Input the private key stored as a binary file that will be understood by the signing component. This option accepts an RSA 2048-bit key (generated from 'openssl genrsa' or similar).

`--cert=file | -r file`

Provide the x509 certificate for signing.

When used with `--sign`, refers to the certificate that corresponds to the given private key.

When used with `--authenticate`, refers to the CA certificate.

`--expand | -e`

The reverse of --compose. The VIB will be unpacked into the optional output directory (by using `--output-dir`), which contains the VIB descriptor file and extracted payload files in the `payload` subdirectory. This directory can be used as input to create a VIB (by using `--compose`).

`--unpack | -u`

This option is deprecated and will be removed in the next release. Use `--expand` instead.

`--info | -i`

Prints information about the input VIB. Typical information includes VIB ID, type, and whether the VIB is signed.

`--offline-depot | -O`

Create a new offline depot with the specified file name from the resulting VIB. This option can only be used with --compose option.