OpenXT™ Synchronizer Administrator Guide High-assurance isolation & security for virtual environments

Publication date 12 June 2014 (build 1550)

Table of Contents

Chapter 1. Introduction

Synchronizer XT is a management center for the OpenXT product, allowing IT administrators to manage and update OpenXT devices, deploy and update virtual machines to them, perform auditing, and take action if the devices become lost or compromised. Synchronizer is a scalable web application, designed to support up to several thousand clients depending on your configuration. Security is paramount for a management application and Synchronizer uses industry-standard techniques to ensure that your devices and management servers remain robust against compromise.

1.1. Basic Operation

Synchronizer XT operates by allowing the administrator to define a target state for a device. This target state consists of configuration, disk data, keys, and upgrade repositories. OpenXT implements the target state defined by the administrator by determining what the desired state is, then taking action to achieve it. The OpenXT device reports its current state to Synchronizer so that the administrator can identify anomalies, view progress, or diagnose problems.

1.2. Components

The interaction between Synchronizer and a managed OpenXT client device is handled by two main components:

- *The Synchronizer client service VM* is a VM that runs on the OpenXT client device, responsible for a single Synchronizer connection.
- *The Synchronizer server side* consists of an Oracle database, a web application that uses the WSGI within Apache, a License Server, and two command-line tools - one to administer the database, the other providing the primary interface to the system.

1.3. Devices, Users and VMs

OpenXT devices can be registered with Synchronizer and configured to connect using a Synchronizer service VM. For a device to connect, it first needs to be authorized on the server. Doing so generates a shared secret that the administrator must place onto the device. This enables mutual authentication between the client and server using HTTP digest authentication over an SSL connection. The client verifies the server certificate prior to initiating communication using standard SSL setup.

OpenXT devices are set up with non-user-specific platform level authentication. Typically user authentication is only handled by the login and lock screens of each virtual machine on the device. The VMs that the Synchronizer downloads will be made available anyone who can unlock the device level authentication.

It is up to the administrator to provide appropriate server-side storage for the VHD files that are the virtual hard disks for the managed VMs. These files can be large in size. The administrator needs to make the storage available to the Synchronizer server at locations that are specified in the database for each virtual disk. The storage may be either local, or mounted from a remote source such as an NFS (this configuration is recommended for higher-volume installations).

If the administrator requires the Synchronizer to operate even when the device is not connected to their local network, then either a port will need to be opened to the Internet so the device can connect when it is roaming, or a VPN VM may be used to provide networking services to the Synchronizer VM and the device configured to route Synchronizer traffic over the VPN.

1.4. Database Access

Different functions of the Synchronizer access the database using different Oracle users to improve the security of the system. There are four Oracle users that would normally be configured:

- <*name*>_owner
- <*name*>_admin
- <*name*>_license
- <*name*>_server

The owner role is for the administrator to get access to the database and has full privilege. The admin role is used by the CLI. The server role is used by the web server and has much more limited privileges. The licensing role is used by the licensing integration dæmon.

You will initially require an appropriate level of access to the database to create the users and schema.

1.5. Virtual disk storage and encryption

Synchronizer stores writes to deployed disks made on devices as a local differences VHD node, separately from the image created by the administrator. The 'base disk' node and the local differences VHD node would normally be stored encrypted, each with its own key. VHD node encryption can be disabled if not required.

Chapter 2. Installing Synchronizer

This chapter describes how to plan, install, and configure the Synchronizer server-side components.

2.1. System requirements

For the client device, hardware supporting OpenXT is required, and OpenXT installed on it, as documented in the *OpenXT Engine Administrator Guide*.

For the server side components, the following needs to be provided:

- *For the database:* An Oracle 11g database needs to be provided. Please see the system requirements described in the Oracle user guide for information on appropriate systems to run this. For many deployments, the free Oracle Express edition will be suitable.
- *Licensing:* Details on licensing schemes are beyond the scope of this document.

  - Windows Server 2003 family
  - Windows Server 2008 family
  - Windows Server 2008 R2 family
  - Windows 7, 32-bit and 64-bit editions

- *For the web server and WSGI scripts:* Red Hat Enterprise Linux 6.3 is recommended to achieve a supported configuration. Although it may be possible to make the system work on other Linux distributions, this is not tested. A 64-bit installation is recommended.
- *For the CLI host:* Red Hat Enterprise Linux 6.3 is recommended.

Any hardware or VM capable of running the above operating systems should be suitable, with a recommended minimum of 4GB of RAM for the server components and a 64-bit server processor.

Note

In a small deployment, it is feasible to run the database, CLI tools, License Server, and Synchronizer server all on a single host.

2.2. Installing the Components

The Synchronizer server-side components are available at the TBD website along with the OpenXT client software. They are RPM packages:

- `sync-database-4.0.0-1.el6.noarch.rpm` - the Synchronizer database schema
- `sync-admin-4.0.0-1.el6.noarch.rpm` - the Synchronizer CLI administration tool
- `sync-server-4.0.0-1.el6.noarch.rpm` - the Synchronizer web server components
- License server package

Also required is the Oracle Python bindings RPM from SourceForge. The link is given below in the installation procedure.

Note

In this version, the RPMs do not have appropriate SELinux labels, so disable SELinux on the database, Synchronizer, and CLI servers, or set the appropriate labels. This is configured in `/etc/sysconfig/selinux`. You will need to reboot after editing this file.

Start by installing the *argparse* Python module on each Synchronizer server that will run the server, admin tool, or database tool, by executing the following command:

```
easy_install argparse
```


Note

You may need to run the command:

```
yum install python-setuptools
```

to install **easy_install**

2.2.1. License Server

Details concerning licensing are beyond the scope of this document.

2.2.2. Oracle Database Server

Install and configure Oracle on the database server. Follow the instructions in the Oracle documentation. If you want to use the free Oracle Express edition, you can download it here[†]. Accept the license agreement, then follow the link for the Linux x64 download.

Note

You will need to log in to your Oracle account, or create one if you do not already have one.

To install and configure Oracle Express edition:

1. Unzip the rpm:

   ```
   unzip oracle-xe-11.2.0-1.0.x86_64.rpm.zip
   ```

2. Install the RPM in the standard way:

   ```
   rpm -ivh oracle-xe-11.2.0-1.0.x86_64.rpm
   ```

3. Configure the database:

   ```
   /etc/init.d/oracle-xe configure
   ```

   following the prompts.

Note

The server's hostname needs to be resolvable in order for this to work. This can be done by ensuring that DNS and the hostname for the machine are configured correctly.

Once the database is installed and configured on the host, install the Oracle Python bindings and the Synchronizer database schema and prepare the database for use with Synchronizer.

---

*

[†]http://www.oracle.com/technetwork/products/express-edition/downloads/index.html

To install the Oracle Python bindings:

1. Download the Oracle Python bindings from SourceForge here[‡].
2. On the database server, install the RPM in the standard way:

```
rpm -ivh cx_Oracle-5.1.2-11g-py26-1.x86_64.rpm
```

3. When the installation is complete, execute the following command to ensure that the library paths are correct for Oracle to use them:

```
echo /u01/app/oracle/product/11.2.0/xe/lib > /etc/ld.so.conf.d/oracle.conf && ldconfig
```

To install the Synchronizer database schema:

1. Download the Synchronizer database schema.
2. On the database server, install the RPM in the standard way:

```
rpm -ivh sync-database-4.0.0-1.el6.noarch.rpm
```

Next, you need to prepare the database by configuring the **sync-database** tool and installing the Synchronizer database schema.

To prepare the database:

1. Ensure that the path to the Oracle binary is in your path by sourcing the `oracle_env.sh` script in your bash profile. For example, create a script called `syncxt.sh` in `/etc/profile.d` with the following contents:

```
. /u01/app/oracle/product/11.2.0/xe/bin/oracle_env.sh
```

2. Run the command:

```
sync-database generate-config ~/.sync-database.conf
```

3. You will be prompted for several parameters, including passwords for the various database users that will be accessing the database. The following is an example; set your *Oracle server* name as appropriate to your environment (if you are running the Oracle database on the same host as the Synchronizer, localhost is the default if you leave it blank) and provide your own passwords at each prompt.

```
Oracle server (blank for localhost): carter.cam.xci-test.com
Password for user 'sys': rootpw
Unique prefix for user names (default sync): sync
Password for user 'sync_owner' (default sync_owner): rootpw1
Password for user 'sync_admin' (default sync_admin): rootpw2
Password for user 'sync_license' (default sync_license): rootpw3
Password for user 'sync_server' (default sync_server): rootpw4

Configuration written to '/home/hsimpson/.sync-database.conf'.
```

4. Once the database configuration has been created, the schema can be installed as follows:

```
sync-database install
```

5. Since many objects used by Synchronizer are identified by UUIDs, it can be cumbersome to use without assistance, so it is best to enable tab completion. Although support for the `/etc/bash_completion.d` directory is not standard in Red Hat Enterprise Linux 6, sourcing the installed completion file either in the terminal or in your `.bashrc` file will enable you to complete UUIDs and commands using the **Tab** key:

```
. /etc/bash_completion.d/sync-database
```

2.2.3. Synchronizer Server

You need to install and configure the sync-admin CLI tool and any license server software on the Synchronizer server.

[‡] http://prdownloads.sourceforge.net/cx-oracle/cx_Oracle-5.1.2-11g-py26-1.x86_64.rpm

To install and configure the sync-admin tool:

1. On the Synchronizer server, install the RPM in the standard way:

```
rpm -ivh sync-admin-4.0.0-1.el6.noarch.rpm
```

2. The sync-admin tool connects to the database for most of its operation. It is possible to pass in details of the database connection on each invocation using the `-d` option, or a configuration file can be used to persist these details. The configuration file for sync-admin is `~/.sync-admin.conf` and would have contents similar to the following example:

Warning

Be certain to supply the correct password for the `sync_admin` user.

```
[database]
login = sync_admin/rootpw
```

3. Enable tab completion for the sync-admin tool:

```
. /etc/bash_completion.d/sync-admin
```

Also add this to your bach profile.

The Synchronizer server also needs to have a license server installed. **To**

install and configure the license server:

1. `This step is beyond the scope of this document.`

2.2.4. Web Server

Next, install the Apache web server (if needed) and its components on the Synchronizer server. If you installed Red Hat Enterprise Linux 6.3 or higher on the Synchronizer server with the web server option selected, Apache is already installed. If required you can install Apache using the yum command:

```
yum install httpd
```

To prepare the web server and install the web server components:

1. Run **system-config-firewall-tui** and enable `Secure WWW (HTTPS)` to be passed through the firewall. You may need to run the command **yum install system-config-firewall-tui** first.
2. Configure Apache to start at boot:

   ```
   chkconfig httpd on
   ```

3. Install the Apache modules `mod_ssl` and `mod_wsgi`:

   ```
   yum install httpd mod_ssl mod_wsgi
   ```

4. Install the sync-server RPM onto the Synchronizer server:

   ```
   rpm -ivh sync-server-4.0.0-1.el6.noarch.rpm
   ```

5. Install certificates onto the host according to the SSL configuration in your configuration file. For testing purposes you may choose to rely on a self-signed certificate. Click here[4] for comprehensive instructions.
6. Create an Apache configuration file. You may choose to modify and use the example given at the end of this step for a proof-of-concept installation. This configuration will need to:

   - Load the WSGI module. The relevant lines in the configuration file are:

     ```
     WSGIPythonPath /usr/lib/python2.6/site-packages/sync_server/scripts
     ```

   - Use SSL and point to the certificate and key file created in the previous step. Ensure that `SSLCertificateFile` and `SSLCertificateKeyfile` contain the correct paths to the installed certificates. The relevant lines in the configuration file are:

     ```
     SSLEngine on
     ```

     and

     ```
     SSLCertificateFile /etc/httpd/conf/sync.crt
     SSLCertificateKeyfile /etc/httpd/conf/sync.key
     ```

   - Listen on port 443 (unless you intend to override this on the client):

     ```
     Listen 443
     ```

   - Set the *WSGIPythonPath* to `/usr/lib/python2.6/site-packages/sync_server/scripts`:

     ```
     WSGIPythonPath /usr/lib/python2.6/site-packages/sync_server/scripts
     ```

   - Configure digest authentication:

     ```
     <Location />
       AuthType Digest
       AuthName "Synchronizer"
       AuthDigestProvider wsgi
       WSGIAuthUserScript /usr/lib/python2.6/site-packages/sync_server/scripts/auth_wsgi.py
       Require valid-user
     </Location>
     ```

   - Have *WSGIScriptAlias* statements so that **/disk** is provided by the script `/usr/lib/python2.6/site-packages/sync_server/scripts/disk_wsgi.py`, **/repo** by the script `/usr/lib/`

[4] http://www.akadia.com/services/ssh_test_certificate.html

python2.6/site-packages/sync_server/scripts/repo_wsgi.py, and / by the script /usr/lib/python2.6/site-packages/sync_server/scripts/server_wsgi.py:

```
WSGIScriptAlias /disk /usr/lib/python2.6/site-packages/sync_server/scripts/disk_wsgi.py
WSGIScriptAlias /repo /usr/lib/python2.6/site-packages/sync_server/scripts/repo_wsgi.py
WSGIScriptAlias / /usr/lib/python2.6/site-packages/sync_server/scripts/server_wsgi.py
```

The following is an example Apache configuration for the purpose of illustration only. It might be insecure or inappropriate for your environment, and should not be used unmodified or without proper review against local security and configuration policies. It will however suffice to modify this example according to the instructions above and use it in its entirety for a proof-of-concept Synchronizer installation.

Your Apache config file needs to be in /etc/httpd/conf and be named httpd.conf. Be sure to save a copy of the default configuration file with another name.

Important

- Ensure that the modules specified in the LoadModule lines are present and install them if not.
- Ensure that your firewall is correctly configured.

```
ErrorLog logs/sync_error_log
LogLevel warn
PidFile run/httpd.pid
DefaultType text/plain

User apache
Group apache

WSGIPythonPath /usr/lib/python2.6/site-packages/sync_server/scripts
LoadModule wsgi_module modules/mod_wsgi.so
LoadModule env_module modules/mod_env.so
LoadModule auth_digest_module modules/mod_auth_digest.so
LoadModule authz_user_module modules/mod_authz_user.so
LoadModule ssl_module modules/mod_ssl.so
LoadModule alias_module modules/mod_alias.so

SSLCertificateFile /etc/httpd/conf/sync.crt
SSLCertificateKeyfile /etc/httpd/conf/sync.key

SSLEngine on
<Location />
  AuthType Digest
  AuthName "Synchronizer"
  AuthDigestProvider wsgi
  WSGIAuthUserScript /usr/lib/python2.6/site-packages/sync_server/scripts/auth_wsgi.py
  Require valid-user
</Location>

WSGIScriptAlias /disk /usr/lib/python2.6/site-packages/sync_server/scripts/disk_wsgi.py
WSGIScriptAlias /repo /usr/lib/python2.6/site-packages/sync_server/scripts/repo_wsgi.py
WSGIScriptAlias / /usr/lib/python2.6/site-packages/sync_server/scripts/server_wsgi.py

Listen 443
```

7. Create the Synchronizer configuration file. This should be named /etc/sync2/sync-*<port>*.conf, where *<port>* is the port that the WSGI scripts are responding on. This port is the one specified in the Apache configuration, for example in the line:

```
Listen 443
```

The file contains ini-file style configuration statements. It has a database section with a login key that contains the login credentials to be used when connecting to the database, and an environment section that defines the environment variables required for the Oracle connection. This is an example of a configuration file:

Warning

Be certain to supply the correct password for the `sync_server` user.

```
[database]
login = sync_server/rootpw

[environment]
ORACLE_HOME = /u01/app/oracle/product/11.2.0/xe
ORACLE_SID = XE
NLS_LANG = ENGLISH_UNITED KINGDOM.AL32UTF8
PATH = /u01/app/oracle/product/11.2.0/xe/bin:$PATH
```

2.3. Upgrading from a previous version of Synchronizer

If you have a previous Synchronizer version, the recommended upgrade order is:

- Temporarily shut down the Apache server
- Upgrade the RPMs
- Upgrade the database
- Restart the Apache server

Note

Run the **. /etc/bash_completion.d/sync-admin** and **. /etc/bash_completion.d/sync-database** commands to update tab completion for any new or altered commands, or start a new shell.

The new server version can be used to deploy an over-the-air upgrade to the clients with new software. Do not use new Synchronizer features such as deploying ISOs until after the upgrade of the clients has completed as these features are not compatible with the older client software.

Upgrading the database:

1. Assuming that the web servers have been stopped, and new RPMs have been installed, use the **sync-database version** to see the state of the database:

   ```
   sync-database version
   Target schema version: 2
   Database schema version: 1

   To upgrade database to schema version 2, use the 'upgrade' command.
   ```

2. Before upgrading the database, ensure that you have an up-to-date backup of the existing version, as the procedure is not reversible and the new database schema is incompatible with the older Synchronizer software.
3. To upgrade the database, run the following command:

   ```
   sync-database upgrade
   ```

4. You may now restart other services.

Note

Schema versions 1 and 2 had the concept of users which were removed in schema version 3. The database upgrade will fail if multiple instances of the same VM owned by different users are assigned to the same device. In such cases it is necessary to edit the database to resolve such conflicts, for instance by deleting all VM instances.

Chapter 3. Managing OpenXT Devices and VMs With Synchronizer

This chapter describes how to configure OpenXT devices to interact with Synchronizer, use its remote management features and deploy VMs.

3.1. Licensing

The specifics of a particular licensing solutions are beyond the scope of this document. This section describes how the XT Synchronizer would interact with a given licensing solution.

When a particular OpenXT device is unlicensed, it displays a warning banner to indicate that it is in an unlicensed state.

When a new OpenXT device is added, it makes an asynchronous request for a license. Until that request is completed, the device is considered unlicensed for a period of time. This is a temporary state, and the device will return to normal when the device next communicates with Synchronizer after a license has been checked out for it.

When a OpenXT device is deleted, the license for the device will be checked back into the license server when the next license update occurs.

If a OpenXT device becomes dormant (that is, it has stopped contacting Synchronizer), at some point its license will be checked back into the license server. This is because licenses are chosen at random periodically. This ensures that the state recorded in the database about the currently-licensed devices (which could easily be edited by an administrator) is consistent with the available licenses. When the OpenXT device next contacts Synchronizer, a license is re-acquired. However, if the device never contacts Synchronizer again, it is assumed to have been removed and its license will be returned to the license server.

3.2. Preparing a OpenXT device to be managed by Synchronizer

The OpenXT platform can be managed by a Synchronizer service VM. The software required to do this is made available when the platform is installed, however in order to use the facility a service VM must be instantiated for each Synchronizer deployment that is to manage the device or VMs on the device.

To configure a OpenXT device to be used with Synchronizer:

1. Create an identifier for the device on the server with the following command. It will return the UUID of the added device:

   ```
   sync-admin add-device mydev
   device_uuid: 9145c4c5-5ab1-4068-8fbf-408eca0660c2
   ```

2. Using this UUID, get the device secret:

   ```
   sync-admin show-device-secret 9145c4c5-5ab1-4068-8fbf-408eca0660c2
   shared_secret: 1b5f6826e082ad3cae0256db3b067e18
   ```

3. On the client device, create a Synchronizer service VM. To do this, you will need to use the command-line. Press **Ctrl + Shift + T** to open a terminal. You might also need to assume appropriate privileges using the **newrole** command:

   ```
   newrole -r sysadm_r
   Password: ************
   xec create-vm-with-template new-vm-sync
   ```

   The return value of the **create-vm-with-template** is the VM object path, which is a long string that looks like **/vm/bf7ab6f6_5c65_43a9_a936_032dcd37256c**.

4. Using this object path, assign a short name to the VM so that it's easier to refer to in later commands.

```
xec -o <vm_object> set name <vm_name>
```

This allows you to execute the xec commands that follow with the syntax **xec -n <*vm_name*>** rather than **xec -0 <*vm_object*>**

5. Create the following directories:

```
mkdir -p /storage/sync/<synchronizer_name>
mkdir -p /storage/sync/<synchronizer_name>/disks
mkdir -p /storage/sync/<synchronizer_name>/repo
mkdir -p /storage/sync/<synchronizer_name>/repo-download
mkdir -p /config/sync/<synchronizer_name>
```

6. Configure the Synchronizer service VM to have access to these two directories:

```
xec-vm -n syncvm set icbinn-path /storage/sync/<synchronizer_name>\
,/config/sync/<synchronizer_name>
```

7. Create the following symlink to prepare the Synchronizer service VM for future upgrades:

```
ln -s /storage/sync/<synchronizer_name>/repo /storage/update-staging
```

8. Ensure the directories created above get the right file context:

```
restorecon -r /config/sync
restorecon -r /storage
```

9. Configuration of the Synchronizer client code is done through domstore. See the configuration values in the table below. Set these using

```
xec-vm -n <synchronizer_service_vm_name> set-domstore-key <key> <value>
```

Key	Value
name	A name for the Synchronizer. Use the name <*synchronizer_name*> used for creating directories on the client device earlier.
url	Location of the Synchronizer. Usually uses https://.
cacert	The CA certificate or bundle to validate the server certificate against. This needs to be the contents of the certificate file, not a path. To accomplish this, copy the CA certificate to `/tmp/cacert` (the `/tmp` directory is one of the few directories in the control domain filesystem that is not readonly), and set the key's value with `xec-vm -n syncvm set-domstore-key cacert "`cat /tmp/cacert/<certname.crt>`"`
device-uuid	The UUID of this device in the Synchronizer database on the Synchronizer server. You can list the devices that exist in the database by runing the **sync-admin list-devices** command.
secret	The shared secret for the device. Use `sync-admin show-device-secret <uuid>` on the Synchronizer server to get this.
interval	Polling interval in seconds. An interval of 15 minutes (900 s) is a reasonable setting. An interval of 24 hours (86400 s) would not be unreasonable if you want to reduce load or network traffic. The shorter the interval, the faster synchronization will start after changes on the server. But be sure to set it no shorter than 1 minute (60 s).

Key	Value
timeout	Time in seconds after which an attempt to reach target state will be aborted ready to retry at the next polling interval (doesn't restart from the beginning). A timeout of 60 minutes (3600 s) with a reliable network and server is a reasonable value. Decreasing for use with an unreliable server is a good idea, if the interval is of comparable length. 1 minute (60 s) or less will slow performance and increase server load.
role	(Optional) Whether the Synchronizer takes the role of a *platform* or a *realm* Synchronizer. A realm synchronizer can create VMs and modify or delete its own VMs. A platform synchronizer can do the same but can also modify the configuration of the devices. If not set, it defaults to *platform*. See Section 3.9: "Multiple Synchronizers and realms" for more on platform and realm Synchronizers.
network/mode	(Optional) Either *dhcp* or *static*. If not set, defaults to *dhcp*. If set to *static*, the following keys are required.
network/address	IP address; required if *network/mode* has been set to *static*.
network/netmask	Netmask; required if *network/mode* has been set to *static*.
network/gateway	IP address of gateway; required if *network/mode* has been set to *static*.
network/dns	IP address of DNS server; required if *network/mode* has been set to *static*.

10. Ensure that the clock is set correctly on the client.
11. Start the Synchronizer service vm using

```
xec-vm -n <vm_name> start
```

Note that once you manually start it after the initial setup and configuration as described here, the Sync service VM will be started automatically on reboots.

3.3. Deploying host configuration

Host configuration is set per-device, either when the device is initially added to the database, or subsequently if modified. The **modify-device-config** command is used to change device configuration subsequently. Typically in this case the configuration is added to, although it can be replaced (that is, the old configuration removed, and new configuration added) if the **-r** or **--replace** options are passed to the CLI.

Configuration is set as three-tuples: *dæmon*, *key*, and *value*. Dæmon is an identifier that helps the Synchronizer service on the client device modify configuration for the correct part of the system or object, Key is the name of the property to change, and Value is the new value of the property. Refer to Chapter 4: "*Using the sync-admin CLI tool*" for more details.

Warning

If configuration is set incorrectly, the Synchronizer service VM may fail to reach target state. This will be reported in the current device state, so it is prudent to verify that changes are successfully deployed, or do a trial run on a test device first.

Please refer to the 'Platform configuration options' for a list of commonly-needed configuration options that can be set. As an example, the following command would disable local VM creation for the device:

```
sync-admin modify-device-config 02d5930e-7622-4f1f-8702-f847e184ce8b -c xenmgr:vm-creation-allowed:false
```

Configuration options not set by Synchronizer will either remain at their default value, or (if not disabled by policy) may be configured by the user.

To see what configuration options are set on a device, the **show-device-config** command can be used:

```
sync-admin show-device-config 02d5930e-7622-4f1f-8702-f847e184ce8b
daemon: xenmgr
key:    vm-creation-allowed
value:  false
```

3.4. Authoring VHD files

Virtual Hard Disks (VHDs) are used similarly on both the client and server as the on-disk storage format for disk images. A VHD file can be encrypted or unencrypted, and if it is encrypted then a separate key file is used, the hash of which is stored in the VHD file.

Warning

VHD files contain metadata that help the system locate them when needed by other files, such as delta disks. One should be careful copying around VHD files and using an existing VHD file as the basis for a new base disk, as this may lead to multiple VHDs with the same "identity" being deployed to a client. Synchronizer will ensure that these conflicts are avoided, but when manually handling VHD files, this must be taken into consideration.

Once a disk has been placed onto the Synchronizer store and referenced from the database (typically by calling the **add-disk** command), the administrator should not change its contents. If an updated version of the VHD file is needed for deployment, a new disk row should be created, referring to a new on-disk file.

Typically, authoring a VHD file is done on OpenXT. The appropriate operating system should be installed, along with OpenXT Tools, and any applications and configuration that the user requires.

To author a VHD file:

1. Create a local VM using the **Install VM** button in the OpenXT user interface.
2. Select the appropriate VM type, memory size, disk size, etc., entering appropriate values for fields in this wizard. For more information on this process, please refer to the *OpenXT Engine Administrator Guide*.
3. Install the VM operating system, either using a DVD or CD, or using a local ISO file.
4. Install the OpenXT Tools.
5. Install any configuration or applications that you wish to deploy to the virtual machine.
6. Shut down the virtual machine.
7. Identify any disks that were used with the VM. This requires an operation at the command line:

    a. Press **Ctrl + Shift + T** to open a command terminal. You may have to assume an appropriate SELinux role to proceed, using the **newrole** command. This is to enhance security.

    ```
    newrole -r sysadm_r
    Password: ***********
    ```

    b. To identify the VM, use the **xec-vm** command:

    ```
    xec-vm
     ID   | Name                | UUID                                 | State
    ------+---------------------+--------------------------------------+--------
       1  | ndvm                | 00000000-0000-0000-0000-000000000002 | running
       2  | uivm                | 00000000-0000-0000-0000-000000000001 | running
          | win7                | 74f90830-40a8-4f38-8138-142dbe4f4b13 | stopped
          | syncvm              | 1171ceb1-cd74-413e-ac3f-0e76a1d374f9 | stopped
          | xp                  | 45fd1628-0829-4b4a-8776-0d007d89932e | stopped
    ```

c. In this case, we may wish to, for example, upload the VM named *win7*. We can then get a list of disks that are used by that VM as follows:

```
xec-vm -n win7 list-disks
/vm/74f90830_40a8_4f38_8138_142dbe4f4b13/disk/0
/vm/74f90830_40a8_4f38_8138_142dbe4f4b13/disk/1
```

d. In this example the VM has two disks, one of which is its virtual hard drive, the other the tools CD. To identify the location on storage of the VHD file required, use the **get phys-path** command for each disk:

```
xec-vm -n win7 --disk 0 get phys-path
/storage/isos/xc-tools.iso
xec-vm -n win7 --disk 1 get phys-path
/storage/disks/f7d8b266-2a67-4348-9461-a1ff2a61ae31.vhd
```

In this case we want to copy the VHD file corresponding to disk 1.

e. Use the **scp** command (or an alternative method) to copy the VHD file from the host to the server, for example:

```
scp /storage/disks/f7d8b266-2a67-4348-9461-a1ff2a61ae31.vhd user@sync:/storage/disks/
```


Note

The directory `/storage/disks` does not exist by default and must be created. Any directory can be used for disk storage.

Note

The location on the server will vary. *<user>* is the username of an appropriate user on the server, and *<sync>* is the hostname of a server with access to the VHD storage you have provisioned for Synchronizer. In a simple configuration, this may be one of the Synchronizer web servers.

f. If the disk is encrypted, you will need to use the key later as a parameter to the **sync-admin** command to add the disk to the database. To determine if the VHD is encrypted, use the **vhd-util key -p -n** command on the client device. Note that the **vhd-util** command can destroy data if used incorrectly.

```
vhd-util key -p -n <filename>
```

If a hash is returned, then the key file would be in the directory `/config/platform-crypto-keys` with a similar filename to the VHD. Copy this file to your Synchronizer.

8. Close the terminal once the operation is complete. Note that leaving a system unattended with an open terminal is a security risk.

3.5. Authoring based on an existing VM

To author based on an existing VM, the same steps as above can be followed assuming that the VM already exists. If the VM was previously provisioned by Synchronizer, or snapshotting had been used, it may be necessary to coalesce the disk image to a single VHD file first.

3.6. Adding VMs to Synchronizer

This section describes how to add VMs to Synchronizer for deployment to clients.

3.6.1. Adding the Disk to Synchronizer

On the Synchronizer CLI host, you can use the **add-disk** command to add a reference to the new VHD file to the server. A hash of the disk is computed unless you specify one manually using the **--hash** option.

To add a reference to the disk so you can use it when deploying a virtual machine later, use the following command on the server:

```
sync-admin add-disk <name> <file-path>
```

<name> is a helpful name that you can use to help identify the disk later. If the disk was encrypted and you would like Synchronizer to deploy the key material to devices when the disk is downloaded, the **--key-file** option must be specified and the encryption key given. Alternatively, the raw key can be passed using the **-k** option, although this takes a hex-encoded string and may result in sensitive data being written to the shell history file.

<file-path> is the location of the file that the web server will deploy.

Each of these commands, when completed successfully, will return the UUID (i.e. identifier) of the disk in the database. You can use this later to refer to the disk.

Note

The disk can be of either VHD or ISO format. ISOs may be attached to VMs in order to install an operating system or other software. If a blank VHD is used as the first or system disk, then the default boot order will cause the VM to fall back to booting from the ISO until an OS is installed. This feature can be used alongside an OS installer configured to auto-install to the disk.

3.6.2. Shared vs. Non-shared Disks

It is possible to "upgrade" a VM on a client device by creating a new VM instance with a new version of the base disk. In the case the local changes are stored on a separate disk, through manual configuration or by using a tool such as Personal vDisk (this is not yet supported for use on Synchronizer XT), it may be desirable for these local changes to be migrated to the new virtual machine. The **shared** flag causes the existing local changes on the device to be used for the disk attached to the new VM instance, rather than creating a separate empty local-changes file.

The **shared** flag is configured on the Synchronizer when the disk is created, and is specified using the **-s** option:

```
sync-admin add-disk -s <name> <file-path>
```

The value of the shared flag is shown in the output of the **list-disks** command.

3.6.3. Disk Encryption

The local changes to a base disk are stored in encrypted form on the client. The encryption key used will be generated on the client and is unique to that device's changes. If you prefer, you can instead opt not to use encryption by setting the **disk:encrypt_snapshots:false** configuration parameter for the VM.

3.6.4. Creating a VM on Synchronizer and Deploying it to a Client

Once the disks have been recorded in the Synchronizer database, a VM object needs to be created that collects them and the configuration for the target VM instances together. There is a single VM object for all VM instances. To deploy a VM to a device, an instance of that VM needs to be created.

Deploying a VM to a Device:

1. Create a VM using the **add-vm** command:

   ```
   sync-admin add-vm -d <disk1-uuid> -d <disk2-uuid> ... \
        -c <config-item> ... \
        <vm-name>
   ```

   Either specify a built in VM template with for instance **-c vmparam:template:new-vm** or include all necessary VM properties including v4v rules.

This will return the UUID of the VM that is created.

2. Deploy the VM to device:

```
sync-admin add-vm-instance <device-uuid> <vm-uuid> <name>
```

When the above procedure is followed, the device should start downloading and installing the VM when it next checks in with Synchronizer according to its polling interval.

Although Synchronizer does not track versions of VMs, often a VM may be intended to replace a previous VM, if security or application updates are added and the administrator wishes to deploy these. Using the VM name field to represent the contents and relationships between VMs is therefore useful for later reference.

3.6.5. Deployment using SCCM or other tools

In order to provide ongoing in-guest management, it may be best in some cases to use Synchronizer to deploy VMs that are subsequently installed and managed using an external tool such as SCCM. In this case, the above steps can be followed, except instead of preparing a VM for execution by installing an OS, only the disk configuration and encryption setup must be done. This will generate empty base disks that can be used alongside a configuration to boot the deployed VM from the network (possibly through a configured VPN VM).

Please refer to the configuration options relating to VM boot order.

3.7. Modifying and Removing Deployed VMs

This section lists commands that can be used to change VM instances that have already been deployed to devices from the Synchronizer command-line.

To Force-Shutdown and Delete a VM:

- Run the command:

```
sync-admin purge-vm-instance <vm_instance_uuid>
```

Warning

This is a disruptive operation. Where appropriate use the **remove-vm-instance** command to allow the user to take steps to salvage data from the device or seek assistance. This command also removes all records of the VM being on the device from the Synchronizer database.

To Delete a VM allowing Graceful Shutdown First:

- Run the command:

```
sync-admin remove-vm-instance <vm_instance_uuid>
```

This command does not forcibly shut down the target VM. The VM is removed when the client polls the Synchronizer server and the VM is shutdown. The VM icon in UIVM for OpenXT will be greyed out as soon as the client polls the Synchronizer server if the VM is not shutdown. Once shutdown, the VM will be listed as locked in UIVM for OpenXT and will not be able to be started. The shutdown VM will be deleted the next time that the client polls the Synchronizer server. To fully remove the VM at a later date, including all records of it in the Synchronizer database, use the **purge-vm-instance** command

The intention is that the administrator can use some out of band mechanism (e.g. email or SCCM) to ensure VM shutdown.

Note

Specify the `--hard` to forcefully shutdown the VM.

To Change the Name of a Deployed VM:

- Run the command:

```
sync-admin modify-vm-instance-name <vm_instance_uuid> <new_vm_name>
```

The new name will be applied when the client polls the Synchronizer server. It is safe to do this to a running VM.

To Modify a Configuration Item of a Deployed VM:

- Run the command:

```
sync-admin modify-vm-config <vm_uuid> -c <config_tuple>
```

This command affects changes of all VMs with the specified `vm_uuid`. Changes will not generally take effect until after the client polls the Synchronizer server and then VM has been restarted. It is safe to do this to a running VM.

To Remove all Configuration from a Deployed VM:

- Run the command:

```
sync-admin modify-vm-config <vm_uuid> -r
```

This command affects changes of all VMs with the specified `vm_uuid`. Changes will not generally take effect until after the client polls the Synchronizer server and then VM has been restarted. It is safe to do this to a running VM.

To Specify New Configuration from a Deployed VM:

- Run the command:

```
sync-admin modify-vm-config <vm_uuid> -f <configuration_file_location>
```

This command affects changes of all VMs with the specified `vm_uuid`. Changes will not generally take effect until after the client polls the Synchronizer server and then VM has been restarted. It is safe to do this to a running VM.

3.8. Configuring Over-the-Air upgrade

Synchronizer can be used to keep your client devices up-to-date by deploying platform updates to them. This functionality will only take effect if Synchronizer is in platform mode, otherwise the configuration will be ignored. Each release of the client software comes with an update image that can be used with over-the-air upgrade.

Adding a repository to Synchronizer:

1. Place the upgrade image in a location accessible to the Synchronizer servers, as would be done for a VHD file as discussed above.
2. Add a reference to the repository to the database:

```
sync-admin add-repo <path-to-file>
```

The update repository is a file ending in .tar that stores the installation files for the software.

Upgrading a device remotely:

1. Identify the UUID of the repository containing the software version to be installed on the device. You can use the **list-repos** command to get a list of repositories that are available, e.g.:

```
sync-admin list-repos
repo_uuid: c9f82084-8b68-4dc8-a07e-b2874b209fe4
release:   3.1.0
build:     130503
file_path: /storage/ota/ota-update.tar
```

2. Use the **sync-admin modify-device-repo** command to set the device to upgrade to the software version provided by the selected repository. The repository UUID is specified using the **-r** option.

When using the **modify-device-repo** command, if the **-r** option is omitted, the repository to be used for the device is cleared, and the device will not attempt to perform an upgrade.

The device will upgrade to the software packaged in the specified repository if the repository contains a version that is newer and that the device can upgrade to from its current version. Some versions of the device software can only be upgraded to from a set of starting versions, so please check the user guide for the version you intend to install to ensure it can be upgraded to from the base version of the device.

3.9. Multiple Synchronizers and realms

A device can be connected to multiple Synchronizers. This feature can be used to provide multiple management domains. The **role** option in the Synchronizer client configuration allows the administrator to configure a Synchronizer with the capability to modify all platform configuration, or only to deploy new VMs. Each Synchronizer can only modify or remove the VMs that it placed onto the system.

3.10. Synchronizer Service VM Networking

There are two aspects to the networking configuration of a Synchronizer Service VM:

1. Which network the Synchronizer Service VM network interface card is connected to.
2. Whether the Synchronizer Service VM should contact a DHCP server or use a static IP address.

OpenXT provides several networks as standard, including:

- `/wired/0/bridged`: bridged access to the wired network
- `/wired/0/shared`: NATted access to the wired network
- `/wifi/0/shared`: NATted access to the wireless network
- `/any/0`: NATted access to either the wired or wireless network - whichever is available

By default, a Synchronizer Service VM is attached to the `/any/0` network and obtains an IP address from the DHCP server running in the Network Driver Service VM, allowing it to contact the Synchronizer server whenever a network connection is available. To view the current network configuration run the following command in the Synchronizer Service VM

```
xec-vm -n <syncvm_name> --nic 0 getall
```

To change the network that a Synchronizer Service VM attaches to:

```
xec-vm -n <syncvm_name> --nic 0 set network <network_name>
```

These domstore keys are optional and should only be set when connecting a Synchronizer Service VM to a different network where it makes sense to use a static IP address.

3.11. Troubleshooting

Logs from the Synchronizer client VM are written to the domain 0 log files. Check `/var/log/messages`.

Troubleshooting:

1. Check the validity of the domstore entries on the Synchronizer Service VM.
2. Check that all required RPMs are installed. (**rpm -q <*part_of_package_name*>**).
3. Check that all required Apache modules are installed (**ls /etc/httpd/modules**).
4. Check that the Synchronizer Service VM can communicate with the server over HTTPS. On the server run:

   ```
   sync-admin list-devices
   sync-admin show-device-secret <device_uuid>
   ```

   On the client device run:

   ```
   xec-vm -n <sync_service_vm_name> get-domstore-key device-uuid
   xec-vm -n <sync_service_vm_name> get-domstore-key secret
   ```

   Verify that the device-uuid and secret match the values shown on the Synchronizer server.

   On the client device run:

   ```
   sshv4v <sync_service_vm_name>
   scp <user>@<synchronizer_host>:<path_to_server_certificate> /tmp
   curl --cacert /tmp/<certificate_name> --digest -u <device_uuid>:<secret> \
   https://<synchronizer_host_fqdn>/hello/1
   ```

   View the Apache logs should you encounter an error.
5. Ensure that the time and date are set appropriately on both the client and the server.
6. Verify that the start date of the server certificate is reasonable.

Chapter 4. Using the sync-admin CLI tool

The **sync-admin** CLI tool is the primary method of interacting with Synchronizer. It has many subcommands, which are listed by running it with the **-h** option.

4.1. Basic Usage

The basic syntax of **sync-admin** is as follows:

```
sync-admin <command> <options to command>
```

All commands support the **-h** option, which provides help for the command. This would describe the required arguments for the command, a description of its behavior, and any options available.

The **sync-admin** tool reads database connection details from its configuration file in **~/.sync-admin.conf** (as described in the setup procedure). You can override database login details using the **-d**, by specifying a string in the format of **-d** *<username>*/*<password>*@*<hostname>*.

4.1.1. Configuration item files

Many commands allow *configuration items* to be specified: these are typically specified using the **-c** option. See below for further details on the format of configuration items and the possible values that may be used.

In cases where many configuration items are required, most commands will also take a parameter specifying a file containing a list of these items. This can be useful to store standard VM configurations out-of-band for use either manually or in scripts. Such a file specifying configuration items would be one or more lines in length, and each line would contain a string that would normally be passed as a single configuration item to the **-c** option. For example:

```
vm:name-description:Linux VM including updated kernel
nic/0:network:/wired/0/shared
```

4.2. Configuration Items

A configuration item is used to configure a global or object-specific property on OpenXT. These are used to set specific behaviors such as the wallpaper, policies, VM name, and so on. Configuration items are specified as a three-tuple, consisting of dæmon, key, and value. The key is the name of a property to set, and value is the target value for the property. The value of the dæmon field determines which object the key will operate on, and these are Synchronizer-specific values, whereas the key is simply a property exposed by the client on the object that is identified.

4.2.1. Dæmon Values and Their Effect for Device Properties

xenmgr	Properties of the com.citrix.xenclient.xenmgr.config interface. Examples include: **autostart** (whether VM autostart is enabled), **vm-creation-allowed** (whether the user can create local VMs), **vm-deletion-allowed** (whether the user can delete VMs), **ota-upgrades-allowed** (whether the user can initiate an over-the-air upgrade of the device software), **measure-fail-action**, **enable-ssh**, **enable-dom0-networking**, **switcher-enabled**, **switcher-keyboard-follows-mouse**, **switcher-resistance**, **language**.
ui	Properties of the com.citrix.xenclient.xenmgr.config.ui interface. Examples include: **wallpaper**, which is a path to the wallpaper to use.

4.2.2. Dæmon Values and Their Effect on VM Properties

vm	Properties of the `com.citrix.xenclient.xenmgr.vm` interface for the VM object on the client, corresponding to the VM instance that the VM configuration belongs to. There are many settings and policies available here, which can be listed on a client using a command such as the following, for the UIVM, and looking at the appropriate section. `xec-vm -n uivm getall` This interface also has the key **encrypted_disks** that specifies if VM user data is stored in encrypted form on the client.
vmparam	VM parameters which can be set in the VM template but are not properties of the VM interface, such as **ui-selectable**. Also, a VM template can be specified this way using the key **template** with a value such as **new-vm**.
disk	Properties of the `com.citrix.xenclient.vmdisk` interface for the disk object.
nic/0, nic/1 etc	Properties of a virtual network interface for the VM. Use key **network** to set the OpenXT network that the virtual network interface should be connected to.
domstore	Set domstore entries for the VM.
rpc	Allows a RPC firewall rule to be set. The key should be the RPC firewall rule with colon replaced by comma, and the value should be **true**. For example **rpc:allow,destination,org.freedesktop.Dbus, interface,org.freedesktop.Dbus.Properties,member, Get:true** allows access to DBus properties.
v4v	Allows a V4V firewall rule to be set. The value should be **true**. For example **v4v:myself->0,80:true** allows access to V4V port 80 in the control domain.
pci	Experimental feature to allow PCI passthrough rules to be specified for the VM. The key specifies the PCI passthrough rule, and the value should be **true**. For example **pci:class=0x200:true** passes through NICs to a VM.

4.3. Command reference

Please see the help built into the **sync-admin** tool for comprehensive command-line reference.