XenServer 5.5 API Cheat Sheet

http://blog.gigaspaces.com/2010/10/05/building-your-own-iaas-part-3-centos-machine-image/

The code snippets below does not include error handling and asynchronous API calls.

Logging into Xenserver

```
  Connection connection = new Connection(new URL("http://" + xenserverHostname));
  Session.loginWithPassword( connection,
                             xenserverUsername, xenserverPassword,
                             xenserverApiversion);  //  default is null
  try {

    // xenserver api goes here
 }
 finally {
    Session.logout(connection);
 }
```

Identify the machine to clone

```
  // namelabelToClone = "CentOS_5.3"
  // find a halted real virtual machine that match the specified name label.

  Set<VM> refVMs = VM.getByNameLabel(connection, namelabelToClone);

  if (refVMs.size() == 1) {

     VM vm = refVMs.iterator().next();
     VM.Record record = vm.getRecord(connection);

     if (!record.isATemplate &&
         !record.isControlDomain &&
         record.powerState == Types.VmPowerState.HALTED) {
       return vm;
     }
  }
```

Clone a new machine

```
   // clone machine by name lable
   VM newVM = vm.createClone(getConnection(), namelabel);
   newVM.setNameDescription(
        getConnection(),
        "Cloned " + namelabelToClone + " at " + new Date().toString());
```

Replace the MAC address from a configurable list.

```
// Recycling MAC addresses solves the DHCP IP address pool problem
// macAddresses is a comma delimited configuration string of
// reusable MAC addresses.

// find first virtual network interface
VIF vif = vm.getVIFs(connection).iterator().next();
String macAddress = getVacantMacAddress(connection,macAddresses);
replaceMacAddress(vif,macAddress);

// returns the first configured MAC address that is not in use
private String getVacantMacAddress(Connection connection, String macAddresses) {
   Set<String> usedMacAddresses = new HashSet<String>();
   Map<VIF, Record> allVifs = VIF.getAllRecords(connection);
   for (VIF vif : allVifs.keySet()) {
     VIF.Record vifrecord = allVifs.get(vif);
     usedMacAddresses.add(vifrecord.MAC);
   }
   String vacantMacAddress = null;
   for (String macAddress : macAddresses.split(",")) {
      if (!usedMacAddresses.contains(macAddress)) {
        vacantMacAddress = macAddress;
        break;
      }
   }
   return vacantMacAddress;
}

// updates the specified vif with the new MAC address
private void replaceMacAddress(VIF vif, String newMac) {
   // get virtual network interface details
   VIF.Record vifrecord = vif.getRecord(connection);
   // destroy old virtual network interface
   vif.destroy(connection);
   // create a new virtual network with the modified MAC address
   vifrecord.MAC = newMac;
   vifrecord.MACAutogenerated = false;
   VIF.create(connection, vifrecord);
 }
```

Start machine and wait for IP address

```
   // This code requires XenServer Tools to be installed on the VM

   newVM.start(connection, false /* don't start paused */, false /*don't force*/);
   List<InetAddress> ipAddresses = new LinkedList<InetAddress>();

   while(true) {

      try {

        ipAddresses= getIpAddresses(newVM);
        break;

      }
      catch (com.xensource.xenapi.Types.HandleInvalid e) {

        Thread.sleep(5000);
      }
   }

   // returns a list of IP addresses of the virtual machine
   // requires the VM to have XenServer Tools installed
   List<InetAddress> getIpAddresses(VM machine) {

     List<InetAddress> ipAddresses = new LinkedList<InetAddress>();
     VMGuestMetrics metrics = machine.getGuestMetrics(getConnection());

     Map<String,String> networks = metrics.getNetworks(getConnection());
     for (String key : networks.keySet()) {
       InetAddress ipAddress = InetAddress.getByName(networks.get(key));
       ipAddresses.add(ipAddress);
     }

     return ipAddresses;
   }
```

Start parameterized script, after network interface is properly loaded

```
   // This code uses the ANT org.apache.tools.ant.taskdefs.optional.ssh.SSHExec
   // task to execute remote shell commands over SSH

   String result = ""
   while (!result.contains("ping")) {
     result = runCommand(ipAddress,5000,"echo ping");
   }

   result = runCommand(ipAddress,5000,"~/startSomething.sh "+param1+" "+param2);

 private String runCommand(String ipAddress, long timeoutMilliseconds,
                           String command) {
   final File output = File.createTempFile("sshCommand", ".txt");
   try {
          final SSHExec task = new SSHExec();
          task.setOutput(output);
                // ssh related parameters
          task.setFailonerror(false);
          task.setCommand(command);
          task.setHost(ipAddress);
          task.setTrust(true);
          task.setUsername(this.username);
          task.setPassword(this.password);
          task.setTimeout(timeoutMilliseconds);
          task.execute();
          String response = readFileAsString(output);
          return response;
   }
   finally  { output.delete(); }
  }
```

Destroy machine and its virtual disk

```
// The tricky part in destroying a VM, is to find the disks before terminating
// the machine, and delete the disks after terminating the machine.
// find vm's virtuaL disks

List<VDI> virtualDiskImagesToDelete = new ArrayList<VDI>();
for (VBD vbd : VBD.getAll(connection)) {

   if (vbd.getVM(connection).equals(vm) {
     VDI vdi = vbd.getVDI(connection);
     if (!vdi.getReadOnly(connection)) {
        virtualDiskImagesToDelete.add(vdi);
     }
  }
}


// destroy vm and disks
vm.destroy(connection);


for (VDI vdi : virtualDiskImagesToDelete) {
   vdi.destroy(connection);
}
```