# CIM Authentication for Lockdown Mode

VMware CIM SMASH/Server Management API

## CIM Ticket Authentication

A CIM client must authenticate before it can access data or perform operations on a VMware® ESXi™ host. The client can authenticate in one of the following ways.

- The client can authenticate directly with the CIMOM on the ESXi host by supplying a valid user name and password for an account that is defined on the ESXi host.

- The client can authenticate with a sessionId that the CIMOM accepts in place of the user name and password. The sessionId (called a "ticket") can be obtained by invoking the `AcquireCimServicesTicket()` method on VMware vCenter™ Server.

VMware recommends using CIM ticket authentication for servers managed by vCenter. If the ESXi host is operating in lockdown mode, the CIMOM does not accept new authentication requests from CIM clients. However, the CIMOM will continue to accept a valid ticket obtained from vCenter Server. The ticket must be obtained using the credentials of any user that has administrative privileges on vCenter Server.

The ticket allows a CIM client to operate as if the host were not in lockdown mode, with one exception: The CIM client cannot initiate a reboot of the ESXi host using the CIM Power State Management provider.

For more information about using authentication for security, see http://pubs.vmware.com/vsphere-50/topic/com.vmware.vsphere.security.doc_50/GUID-BD1F9872-9A7F-40BB-93D0-19435F052324.html. For more information about Lockdown Mode, see http://pubs.vmware.com/vsphere-50/topic/com.vmware.vsphere.security.doc_50/GUID-88B24613-E8F9-40D2-B838-225F5FF480FF.html.

## Using the vSphere SDK for Perl to Request a CIM Ticket

Example 1 shows a Perl script using the VMware vSphere SDK for Perl to obtain a ticket that CIM clients can use to authenticate. The script invokes the `AcquireCimServicesTicket()` method of the HostSystem managed object that corresponds to the ESXi host to which the client will connect.

**Example 1.** CIM Ticket Perl Script: cim_ticket.pl

```perl
#!/usr/bin/perl
use strict;
use warnings;
use VMware::VIRuntime;

=pod
  This sample gets a CIM ticket from vCenter Server.

  USAGE:: perl cim_ticket.pl --server vc.example.com --username abc \
          --password xxxx --host esx.example.com
=cut

my %opts = (
    host => {
        type => "=s",
        variable => "VI_HOST",
        help => "Fully qualified name of the ESX host to which the CIM ticket applies",
        required => 1,
}, );
Opts::add_options(%opts);
Opts::parse();
Opts::validate();

Util::connect();

# Get the specified HostSystem object:
my $host = Opts::get_option( 'host' );
my $search_index = Vim::get_view(
   mo_ref => Vim::get_service_content() -> searchIndex );
my $host_moref = $search_index ->
                 FindByDnsName( dnsName => $host,
                               vmSearch => 'false' )
                 || die "Error: Host not found.\n";
my $host_system = Vim::get_view( mo_ref => $host_moref );
# Get the CIM ticket and write it to outfile:
my $ticket = $host_system -> AcquireCimServicesTicket();
print $ticket -> sessionId;

# Disconnect from the server
Util::disconnect();
```

The following command shows how you can use the script in Example 1 to get a ticket from vCenter Server:

```
$ perl cim_ticket.pl --server=vc.mydomain.com --username=administrator \
--password=secret --host=esx.mydomain.com
```

The user name can be any vCenter user with administrative privileges. The host name of the ESXi server must be fully qualified to match the name property of the HostSystem managed object.

## Using the CIM Ticket to Authenticate

You can pipe the output of `cim_ticket.pl` into your CIM client, or you can save it to a text file. Use the ticket in place of both the user name and the password when you authenticate with the CIMOM. The CIMOM responds as if the host were not in lockdown mode.

A ticket stored in a text file can be re-used, but eventually the ticket will time out. To avoid a ticket time-out, you can continue to request a new CIM ticket from vCenter for each connection to an ESXi host.

Initially, a CIM ticket is valid for 120 seconds before it times out. If you use it within 120 seconds to authenticate with the CIMOM, the ticket lifetime is extended by 15 minutes. You can continue to extend the ticket lifetime in 15-minute increments by using it within the last 15 minutes before it expires. However, the maximum lifetime for a CIM ticket is 60 minutes. After that, you must request a new CIM ticket from vCenter.

Extensions to the lifetime of a CIM ticket are conditional. After the first 120 seconds of the ticket lifetime, the ticket is valid only as long as it remains in the ticket cache in the SFCB process. SFCB caches only the most recent 15 CIM tickets. Depending on system load, a ticket can expire before its 15-minute extension completes. Your client code must be able to detect an authentication failure and request a new ticket whenever that happens.

## Requesting a Ticket from the CIM Client

You do not need to run a separate script to acquire a CIM ticket. Any CIM client can request a ticket. shows the Perl code to request a ticket, combined with a WSMan client that uses the ticket to authenticate. With this approach, the ticket is not stored in the clear, except in the client's address space.

**Example 2.**  Issuing a Ticket Request from the CIM Client

```perl
#!/usr/bin/perl
use strict;
use warnings;
use VMware::VIRuntime;
use WSMan::StubOps;
use VMware::VILib;

$Util::script_version = "1.0";

=pod
  USAGE:: perl firmwarerevisions.pl --server vc.example.com --username abc
          --password xxxx --host esx.example.com
=cut

my %opts = (
    host => {
        type => "=s",
        variable => "VI_HOST",
        help => "Fully qualified name of the ESX host",
        required => 1,
     },
   namespace  => {
      type      => "=s",
      help      => "Namespace for all queries. Default is :root/cimv2",
      required => 0,
      default => "root/cimv2",
   },
   timeout  => {
      type  => "=s",
      help  => "Default http timeout for all the queries. Default is 120",
      required => 0,
      default => "120"
   },
);
Opts::add_options(%opts);
Opts::parse();
Opts::validate();
```

```perl
# Save credentials for later ticket requests:
my $username = Opts::get_option ('username');
my $password = Opts::get_option ('password');

# WS-Man connection object:
my ($ticket, $client) = '';

display_firmware_revisions();

sub get_ticket {
   # Connect to vCenter Server for CIM ticket, using default WS API options:
   Opts::set_option('protocol', 'https');
   Opts::set_option('servicepath','/sdk/webService');
   Opts::set_option('portnumber', '443');
   Opts::set_option('username', $username );
   Opts::set_option('password', $password );
   Util::connect();

   # Get the specified HostSystem object:
   my $host = Opts::get_option( 'host' );
   my $search_index;
   eval {
      $search_index = Vim::get_view(
         mo_ref => Vim::get_service_content() -> searchIndex );
   };
   if ( $@ ) {
      print "Failed to get searchIndex object from Service Content.\n";
      die $@;
   }
   my $host_system;
   eval {
      $host_system = Vim::get_view(
         mo_ref => $search_index ->
                   FindByDnsName( dnsName => $host,
                                  vmSearch => 'false' ) );
   };
   if ( $@ ) {
      print "Failed to get HostSystem object by name.\n";
      die $@;
   }
   my $ticket_ref;
   eval {
      $ticket_ref = $host_system -> AcquireCimServicesTicket();
   };
   if ( $@ ) {
      print "Failed to get CIM ticket.\n";
      die $@;
   }

   # Disconnect from vCenter Server
   Util::disconnect();
   return $ticket_ref -> sessionId;
}
```

```perl
sub get_wsman_client
{
   my ($ticket) = @_;
   # Connect to CIM server, using default WS-Man options:
   Opts::set_option('protocol', 'http');
   Opts::set_option('servicepath','/wsman');
   Opts::set_option('portnumber', '80');
   my %args = (
       path => Opts::get_option ('servicepath'),
       username => $ticket,
       password => $ticket,
       port => Opts::get_option ('portnumber'),
       address => Opts::get_option ('host'),
       namespace => Opts::get_option('namespace'),
       timeout  => Opts::get_option('timeout')
   );
   # Create client connection object for ESX host.
   $client = WSMan::StubOps->new( %args );
}


sub print_sw_ident
{
        if ( $_->Name ) {
            print "Name : ",$_->Name,"\n";
        }
        else { print "Name: Not Available\n"; }
        if ( $_->Manufacturer ) {
           print "Manufacturer : ",$_->Manufacturer,"\n";
        }
        else { print "Manufacturer : Not Available\n"; }
        if ( $_->VersionString ) {
             print "Version : ",$_->VersionString,"\n";
        }
        else { print "Version : Not Available\n"; }
        if ( $_->ReleaseDate ) {
           $_->ReleaseDate=~m/(.{4})(.{2})(.{2})/;
           print "Release Date : $1\\$2\\$3\n\n";
        }
        else { print "ReleaseDate : Not Available\n\n"; }
}


sub client_call
{
   my ($function_call) = @_;
   # Catch exceptions; retry authentication failures with new ticket
(once only):
   if ( not $ticket ) {
      $ticket = get_ticket();
   }
   if ( not $client ) {
      $client = get_wsman_client( $ticket );
   }
   my @result = eval( '$client->' . $function_call );
   return ( wantarray ? @result : pop @result ) unless $@;
   if ( $@ =~ m/^401 Unauthorized/ ) {
      $ticket = get_ticket();
      $client = get_wsman_client( $ticket );
      @result = eval( '$client->' . $function_call );
      return ( wantarray ? @result : pop @result ) unless $@;
   }
   print "Client call " . $function_call . " failed.\n";
   die $@;
}
```

```
sub display_firmware_revisions
{
   my @SWRevs = client_call( 'EnumerateInstances( class_name =>
"CIM_SoftwareIdentity" )' );
   print "\n";
   if ( scalar( @SWRevs ) ) {
      foreach (@SWRevs) {
         print_sw_ident( $_ );
      }
   }
   else {
      print "No Software Data available on $ARGV[0]:$ARGV[2]/$ARGV[1]\n";
   }
}
```

The following command shows how you can run the script in Example 2:

**$ perl firmwarerevisions.pl --server=vc.mydomain.com --username=administrator \
--password=secret --host=esx.mydomain.com**

The --server parameter indicates the vCenter Server that manages the ESXi host. The --host parameter indicates the ESXi host on which the CIMOM runs.

Some of the options provided by VILib have default values suitable for a Web Services API connection. When the script makes a WS-Man connection to the ESXi host, it changes some of the option values. If your client must reauthenticate with vCenter Server, such as when the CIM ticket expires, the client must set the correct option values for the Web Services connection. Then the client must set the correct option values for the WS-Man connection before making a WS-Man call.

The affected option values for the Web Services connection are:

```
Opts::set_option('protocol', 'https');
Opts::set_option('servicepath','/sdk/webService');
Opts::set_option('portnumber', '443');
```

The affected option values for the WS-Man connection are:

```
Opts::set_option('protocol', 'http');
Opts::set_option('servicepath','/wsman');
Opts::set_option('portnumber', '80');
```