Mission Guide: COM Libraries

Programming the F-Response COM libraries

Using the F-Response COM Libraries to perform one or more actions via scripts or 3rd party development

Overview

F-Response (Enterprise, Consultant + Covert, Consultant, TACTICAL, and Field Kit) all contain a series of Microsoft Component Object Model (COM) libraries that provide the majority of all functionality for F-Response products. The COM libraries include both documented and undocumented methods and properties. We caution you against using any of the undocumented methods as they are not presently used and may represent stubs for future enhancements.

Understanding the F-Response COM Libraries

The F-Response COM libraries consist of four distinct out of process COM servers that run on the examiner machine (or wherever you have installed the full F-Response software package). The COM Servers are:

COM Server	Purpose
F-Response Connector Server	Provides all the objects and methods for interacting with cloud and 3rd party data providers.
F-Response License Server	Provides all the objects and methods for interacting with the license dongles, determining version and expiration, etc.
F-Response Imager Server	Provides all the objects and methods for creating, starting, stopping and managing images.
F-Response Subject Server	Provides all the objects and methods for deploying and connecting to F-Response subjects and targets.

Imager COM Library

The Imager COM library contains a single class called the "FRImager" class. Instantiating that class creates an object capable of performing all the actions offered by the F-Response Imager. The following examples from the Imager User Guide cover the basic process for using the Imager COM library from Powershell.

Creating a Physical Image using COM Scripting

Physical Images of F-Response presented devices can be created directly from any scripting language that supports COM. Included below is a sample script for creating a Physical Image using Powershell. Do **not** run Powershell using an elevated credential (ie Run as Admin).

```
################################################################################
# F-Response Imager Script for imaging a physical device
# Pre-requisites: F-Response Imager
# Author: M. Shannon
# usage, physical-imager-script.ps1 -sourcedevice <\\.\PhysicalDriveX> -destinationpath
<DESTINATIONPATH Ex. E:\> -imagename <IMAGENAME Ex. XYZImage>
# -examinername <EXAMINER Ex. "Joe Examiner"> -desc <DESCRIPTION Ex. "Some Image Desc"> -
casenum <CASENUM Ex. "Case123"> -evidencenum <EVIDENCENUMBER Ex. "Evidence123"> -notes <NOTES
Ex. "Some Notes">
# -hash <Md5=1, Sha1 = 2, Both = 3 Ex. 1> -compression <None = 0, Fast = 1, Normal = 6, High
=9 Ex. 1>
################################################################################
Param(
        [parameter(Mandatory=$true)]$sourcedevice,
        [parameter(Mandatory=$true)]$destinationpath,
        [parameter(Mandatory=$true)]$imagename,
        [parameter(Mandatory=$true)]$examinername,
        [parameter(Mandatory=$true)]$desc,
        [parameter(Mandatory=$true)]$casenum,
        [parameter(Mandatory=$true)]$evidencenum,
        [parameter(Mandatory=$true)]$notes,
        [parameter(Mandatory=$true)]$hash,
        [parameter(Mandatory=$true)]$compression
)

Set-Variable Imaging 1
Set-Variable Complete 2
Set-Variable Error 5

#initialize the com library
$fresponseimager = New-Object -comObject frimagerserver.FRImager

#start the image
$ImageId =
$fresponseimager.CreatePhysicalImage($sourcedevice,$destinationpath,$imagename,$examinername,
$desc,$casenum,$evidencenum,$notes,$hash,$compression)

if($ImageId -eq 0){
        write "Error starting image, check the event logs for further information."
        [System.Runtime.Interopservices.Marshal]::ReleaseComObject($fresponseimager)
        Remove-Variable fresponseimager
        Exit(1)
}

$ImageState = $fresponseimager.ImageState($ImageId)
$ImageLastStatus = $fresponseimager.ImageLastStatus($ImageId)
write "Beginning Image..."
write $ImageLastStatus
```

```
do {
        Start-Sleep -sec 1
        $ImageLastStatusNew = $fresponseimager.ImageLastStatus($ImageId)
        if ($ImageLastStatusNew -ne $ImageLastStatus){
                write $ImageLastStatusNew
                $ImageLastStatus = $ImageLastStatusNew
        }
        $ImageState = $fresponseimager.ImageState($ImageId)
}while($ImageState -eq $Imaging)

if ($ImageState -eq $Complete){
        write "Image Completed Successfully"
}
if ($ImageState -eq $Error){
        write "Image Ended with Error"
}
$fresponseimager.RemoveImage($ImageId)

[System.Runtime.Interopservices.Marshal]::ReleaseComObject($fresponseimager)
Remove-Variable fresponseimager
```

Creating a Logical Image using COM Scripting

Logical Images can be created directly from any scripting language that supports COM. Included below is a sample script for creating a Logical Image using Powershell. Do **not** run Powershell using an elevated credential (ie Run as Admin).

```
################################################################################
# F-Response Imager Script for imaging a logical device (network share)
# Pre-requisites: F-Response Imager
# Author: M. Shannon
# usage, logical-imager-script.ps1 -sourcedevice <X:\> -imagename <IMAGENAME Ex. XYZImage>  -
outformat <E0x = 1, VHD = 2, Both = 3 Ex. 1>
# -destinationpath <DESTINATIONPATH Ex. E:\>
# -examinername <EXAMINER Ex. "Joe Examiner">
# -desc <DESCRIPTION Ex. "Some Image Desc">
# -casenum <CASENUM Ex. "Case123">
# -evidencenum <EVIDENCENUMBER Ex. "Evidence123">
# -notes <NOTES Ex. "Some Notes">
# -hash <Md5=1, Sha1 = 2, Both = 3 Ex. 1>
# -compression <None = 0, Fast = 1, Normal = 6, High =9 Ex. 1>
################################################################################
Param(
        [parameter(Mandatory=$true)]$sourcedevice,
        [parameter(Mandatory=$true)]$imagename,
        [parameter(Mandatory=$true)]$outformat,
        [parameter(Mandatory=$true)]$destinationpath,
        [parameter(Mandatory=$true)]$examinername,
        [parameter(Mandatory=$true)]$desc,
        [parameter(Mandatory=$true)]$casenum,
        [parameter(Mandatory=$true)]$evidencenum,
        [parameter(Mandatory=$true)]$notes,
        [parameter(Mandatory=$true)]$hash,
        [parameter(Mandatory=$true)]$compression
)

Set-Variable Imaging 1
Set-Variable Complete 2
Set-Variable Error 5

#initialize the com library
$fresponseimager = New-Object -comObject frimagerserver.FRImager
```

```
#start the image
$ImageId =
$fresponseimager.CreateLogicalImage($sourcedevice,$imagename,$outformat,$destinationpath,$exa
minername,
$desc,$casenum,$evidencenum,$notes,$hash,$compression)

if($ImageId -eq 0){
        write "Error starting image, check the event logs for further information."
        [System.Runtime.Interopservices.Marshal]::ReleaseComObject($fresponseimager)
        Remove-Variable fresponseimager
        Exit(1)
}

$ImageState = $fresponseimager.ImageState($ImageId)
$ImageLastStatus = $fresponseimager.ImageLastStatus($ImageId)
write "Beginning Image..."
write $ImageLastStatus

do {
        Start-Sleep -sec 10
        $ImageLastStatusNew = $fresponseimager.ImageLastStatus($ImageId)
        if ($ImageLastStatusNew -ne $ImageLastStatus){
                write $ImageLastStatusNew
                $ImageLastStatus = $ImageLastStatusNew
        }
        $ImageState = $fresponseimager.ImageState($ImageId)
}while($ImageState -eq $Imaging)

if ($ImageState -eq $Complete){
        write "Image Completed Successfully"
}
if ($ImageState -eq $Error){
        write "Image Ended with Error"
}
$fresponseimager.RemoveImage($ImageId)

[System.Runtime.Interopservices.Marshal]::ReleaseComObject($fresponseimager)
Remove-Variable fresponseimager
```

Subject and License COM Libraries

The Subject and License COM libraries are responsible for listing active subjects, attaching and detaching remote targets, as well as authenticating to and managing subjects. All interaction with the libraries are handled through the "FRSubjectServer" and "FRLicenseManager" classes. The following example outlines the process in powershell for deploying and connecting to a remote machine.

Deploying and connecting to F-Response on a remote machine (Enterprise and Consultant + Covert)

```
#Initialize the COM Libraries
$SubjectServer = New-Object -comObject subject_ctrl.FRSubjectServer
$LicenseServer = New-Object -comObject license_ctrl.FRLicenseManager

#Deploy the software to a remote machine

#Add a Deployment Credential
#SubjectServer.DeployViaLanWanAddWindowsCredential "test","TEST","testpass"

#Deploy the software
Write-Host "Deploying the F-Response Software to x64-win2k12-sub"
$MachineStatus = $SubjectServer.DeployViaLanWanInstallStart("x64-win2k12-sub")
```

```
if ($MachineStatus -eq 13) {
    Write-Host "F-Response Software Installed"
        #We have a started service on the remote machine
        $Subjects = $LicenseServer.GetListOfSubjects()
        #Walk the Subjects to get our URI
        foreach ($Subject in $Subjects){
                Write-Host "Found Subject $Subject"
         $SubjectName = $LicenseServer.GetSubjectHostname($Subject)
         Write-Host "SubjectName is $SubjectName"
                $SubjectURI = $LicenseServer.GetSubjectAddress($Subject)
                Write-Host "SubjectURI is $SubjectURI"
                $SubjectServer.ScanForTargetsBySubject($SubjectURI)
                $Targets = $SubjectServer.GetSubjectTargets($SubjectURI,"*")
                foreach ($Target in $Targets){
                        $TargetName = $SubjectServer.GetSubjectTargetName($SubjectURI,$Target)
                        Write-Host "Found Target $TargetName"
                        if ($TargetName -eq "disk-1"){
                                $MountedDevice =
$SubjectServer.MountDevice($SubjectURI,$Target)
                                Write-Host "Mounted $TargetName as $MountedDevice"
                                Sleep(10)
                                $SubjectServer.UnMountDevice($SubjectURI,$Target)
                                Write-Host "Unmounted $TargetName"
                        }
                }
    Write-Host "Removing software from $SubjectName"
        $SubjectServer.RemoteStopSubject($SubjectURI)
        }
}
```