Common Management Platform
EMC® Common Object Manager (ECOM) Toolkit

2.7.2.0.0

ECOM Deployment and Configuration Guide

P/N 300-015-003
REV 01

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Contents

Preface

The EMC Common Object Manager (ECOM) is a hub of communications and common services for applications based on EMC's Common Management Platform. This document contains an overview of ECOM, instructions for its deployment and configuration, basic use cases for ECOM, and other related reference material.

As part of an effort to improve and enhance the performance and capabilities of its product lines, EMC periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

Audience

This document is part of the Common Management Platform documentation set, and it is intended for use by EMC developers seeking to create and deploy management applications that rely on ECOM for resource exposure.

Readers of this document are expected to be familiar with the following topics:

- Common Information Model (CIM)
- Web-Based Enterprise Management (WBEM)
- Storage Management Initiative - Specification (SMI-S)
- Service Location Protocol (SLP)
- IP Multicast

Conventions used in this document

EMC uses the following conventions for special notices.

Note: A note presents information that is important, but not hazard-related.

CAUTION

A caution contains information essential to avoid data loss or damage to the system or equipment.

IMPORTANT

An important notice contains information essential to operation of the software.

WARNING

A warning contains information essential to avoid a hazard that can cause severe personal injury, death, or substantial property damage if you ignore the warning.

DANGER

A danger notice contains information essential to avoid a hazard that will cause severe personal injury, death, or substantial property damage if you ignore the message.

Typographical conventions

EMC uses the following type style conventions in this document:

Normal	Used in running (nonprocedural) text for: • Names of interface elements (such as names of windows, dialog boxes, buttons, fields, and menus) • Names of resources, attributes, pools, Boolean expressions, buttons, DQL statements, keywords, clauses, environment variables, filenames, functions, utilities • URLs, pathnames, filenames, directory names, computer names, links, groups, service keys, file systems, notifications
Bold:	Used in running (nonprocedural) text for: • Names of commands, daemons, options, programs, processes, services, applications, utilities, kernels, notifications, system call, man pages

	Used in procedures for: • Names of interface elements (such as names of windows, dialog boxes, buttons, fields, and menus) • What user specifically selects, clicks, presses, or types
Italic:	Used in all text (including procedures) for: • Full titles of publications referenced in text • Emphasis (for example a new term) • Variables
`Courier:`	Used for: • System output, such as an error message or script • URLs, complete paths, filenames, prompts, and syntax when shown outside of running text
`Courier bold:`	Used for: • Specific user input (such as commands)
`Courier italic:`	Used in procedures for: • Variables on command line • User input variables
< >	Angle brackets enclose parameter or variable values supplied by the user
[ ]	Square brackets enclose optional values
|	Vertical bar indicates alternate selections - the bar means “or”
{ }	Braces indicate content that you must specify (that is, x or y or z)
...	Ellipses indicate nonessential information omitted from the example

Where to get help

Support, downloads, and licensing information can be obtained as follows.

Product information — For documentation, release notes, software updates, or for information about this and other products and services, go to:

http://cmpdistributions.eng.emc.com

Technical support — For technical support, go to the forums:

http://cmgportal.lss.emc.com/forum/

Your comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Please send your opinion of this document to:

scribes-CMG@emc.com

1 Overview of CIM and SMI-S

Interoperability within an heterogeneous environment can be achieved by ensuring that management applications and service providers hold the same conceptual view of resources within their domain. ECOM supports the EMC Common Information Model (ECIM), which uses an enterprise wide set of models of specific resource and information elements to represent the varied components found within today's diverse IT environments. ECIM is based on the industry-standard Common Information Model (CIM) and also incorporates SMI-S in addition to EMC-specific models and profiles. A knowledge of CIM and SMI-S is therefore necessary to understand the framework within which management applications and service providers interoperate using ECOM. Within this book, a *service provider* is defined as a managed element that exposes data or functionality to clients on a network for consumption by other service providers or by management applications, which are both grouped in to the category *service client* or *consumer*.

The following sections provide a basic overview of CIM, SMI-S, and the communication protocols needed to support the model-driven framework of the Common Management Platform, including ECOM.

Modeling with CIM and SMI-S

This section provides a brief overview of the structure, use, and representation of CIM-based models.

Structural model overview

The CIM schema provides a common methodology for representing systems, networks, applications, and services as a set of object-oriented models that can be bound to real-world data and functionality. Management applications based on CIM can interact with heterogeneous resource instances, such as data storage hardware from multiple vendors, without direct knowledge of the underlying system interfaces.

CIM defines several basic model structures for IT resources.

Classes

CIM *classes* identify types of resources. A class can represent a broad category of resources or can be subclassed to represent a specific type. For example, the class `CIM_NetworkPort` represents a broad category of heterogeneous network communications hardware, while `EMC_NetworkPort` is a subclass that represents an EMC-specific subset.

While classes define types of things found in a managed IT environment, *instances* represent individual implementations of a class. A specific port at a specific network address is an example of an instance of class `EMC_NetworkPort`. Classes define the state, behavior, and identity information (called *properties*) that distinguishes each instance. CIM also defines the operations that can be performed by instances of a class (called *methods*), linking models to system-specific functionality.

Qualifiers

Many elements in CIM can be further specialized through the use of *qualifiers*. Qualifiers enhance or apply constraints to associations, classes, indications, methods, method parameters, properties and references by providing additional information that distinguishes them from similar elements and/or their superclasses. For example, qualifiers can be used to make a class property vendor-specific by defining minimum and maximum values for it. as appropriate for the vendor’s modeled resource.

Associations

An *association*, which is a class with the association qualifier, defines a relationship between two or more classes. An association contains two or more *references* that define the classes linked by the association. However, because an association is its own separate class, it does not impact the associated classes themselves. For example, an association could link the classes representing the virtual machines defined on a server to the class of the server itself.

Indications

An *indication*, which is a class with the Indication qualifier, defines an event of interest that might occur in a managed environment. Instances of Indications are transient objects and cannot be queried using standard CIM Intrinsics. Rather, to receive indications, an application must create a *subscription* to an indication class to receive notification of its instances. Indications can be divided into two categories. *Lifecycle indications* are notifications related to the creation, deletion, and modification of objects modeled in CIM. *Process indications* are notifications of events that may not be completely modeled by CIM. Direct feedback from instrumented devices or SNMP traps are examples of process indications. Subscribers to indications can define *filters* that further limit the indications they are notified of.

Profiles

Because CIM contains almost 2000 classes and is meant to have broad applicability to managed IT environments, some industry groups haven chosen to create *profiles* that address the usage of CIM in specific domains. To promote device and application interoperability, profiles unambiguously define sets of classes, associations, indications, methods, properties that must be used to model the managed resources of the domain. In addition, profiles also specify required property values, required interconnections for associations, and certain method behaviors that are are not typically encoded into MOF files. Profiles also define other requirements that cannot be represented in MOF, such as the use of a specific communication protocol by devices adhering to the profile. Profiles generally include recipes that serve as examples to guide server-side implementers and client-side application writers in proper usage of the profile.

Compliance with specific profiles ensures that managed resources and management applications share the same conceptual view of their environment and of each other. SMI-S is a good example of profiles applied to a storage-related domain.

SMI-S

The Storage Management Initiative - Specification (SMI-S) published by the Storage Networking Industry Association (SNIA) provides several examples of profiles that are used to enable the interoperability of heterogeneous storage and storage-related resources within a network. SMI-S includes domain profiles and subprofiles for arrays, switches, storage virtualization, volume management, and many other storage-related resources and concepts. Additionally, SMI-S defines:

- the communications protocol (CIM-XML) used to exchange CIM models via XML (xmlCIM) over HTTP(S)
- the protocol used to discover available services, which is called the Service Location Protocol (SLP) version 2

ECOM complies with SMI-S version 1.2, including compliance with the following standards:

- CIM schema version 2.14
- CIM/XML version 2.2
- CIM/XML over HTTP version 1.2
- Service Location Protocol (SLP) version 2
- SSL 3.0 / TLS 1.0

MOF

The Managed Object Format (MOF) is a way to represent CIM models in text form. Within a MOF file, the classes, associations, properties, references, methods and associated qualifiers are defined for a domain to be modeled. MOF files are often the starting point for developers seeking to create providers that expose modeled data and functionality through ECOM. MOF files can used by tools such as the OSL# MOF Compiler to generate C++ header and source files that represent the objects modeled in the MOF file. These generated files form the basis for the resource exposure plugin that links ECOM to resource-specific functionality.

```
[ EMC_Sample2IsLeaf,
EMC_CtorArgTypes{ "const char *", "const char *" },
EMC_CtorArgNames{ "arrayId",      "volumeId"     },
Description
    ( "Sample storage volume" ) ]
class SP_StorageVolume : EMC_StorageVolume
{
};
```

Figure 1 MOF file entry for class SP_StorageVolume

Interoperability with CIM

The consistent modeling of domain resources is only the first step in creating a network of interoperable management applications and managed resources. The abilities to exchange model information, to retrieve data from and execute commands through models, and to discover available resources are also required to link the heterogeneous elements in a network. The following sections describe the communication and discovery requirements of SMI-S 1.2. These requirements are fully supported by ECOM.

WBEM

Web-Based Enterprise Management (WBEM) is a group of Web-based technology standards advocated by the Distributed Management Task Force (DMTF) to facilitate unified management of enterprise IT environments. WBEM incorporates:

- a model (CIM) to represent resources
- an XML-based communication protocol (CIM-XML over HTTP) to facilitate interaction between network components[1]
- an XML representation of CIM models and messages (xmlCIM) to travel via CIM-XML
- a discovery protocol (SLP) to allow management applications to discover available services

The combination of WBEM technologies allows ECOM to serve as the interoperability hub of the Common Management Platform (CMP).

1. WS-Management is also advocated by the DMTF as an alternative to CIM-XML.

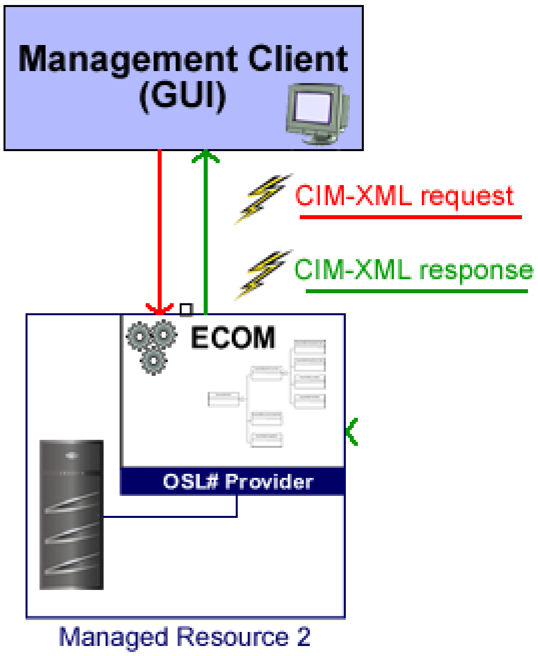

Figure 2 **In this very simple example, a WBEM-enabled client and service provider use the CIM-XML protocol to transmit messages and models between themselves.**

SLP

Before resources can be managed, they must first be discovered by management applications. The Service Location Protocol (SLP) defines a mechanism for using UDP or TCP multicasts to announce the need for and the responding location of services in a network. Applications looking for a service are called *user agents* (UA), and applications providing a service are called *service agents* (SA). optionally, *directory agents* may be used to provide a central listing of available services.

In terms of SLP, ECOM acts as a service agent that advertises its address and capabilities when it receives a multicast requesting its type of available functionality from an user agent (often a management application) on the network. When a UA has found the

ECOM-exposed services that it needs, it begins communicating directly with each ECOM instance via CIM-XML messages.

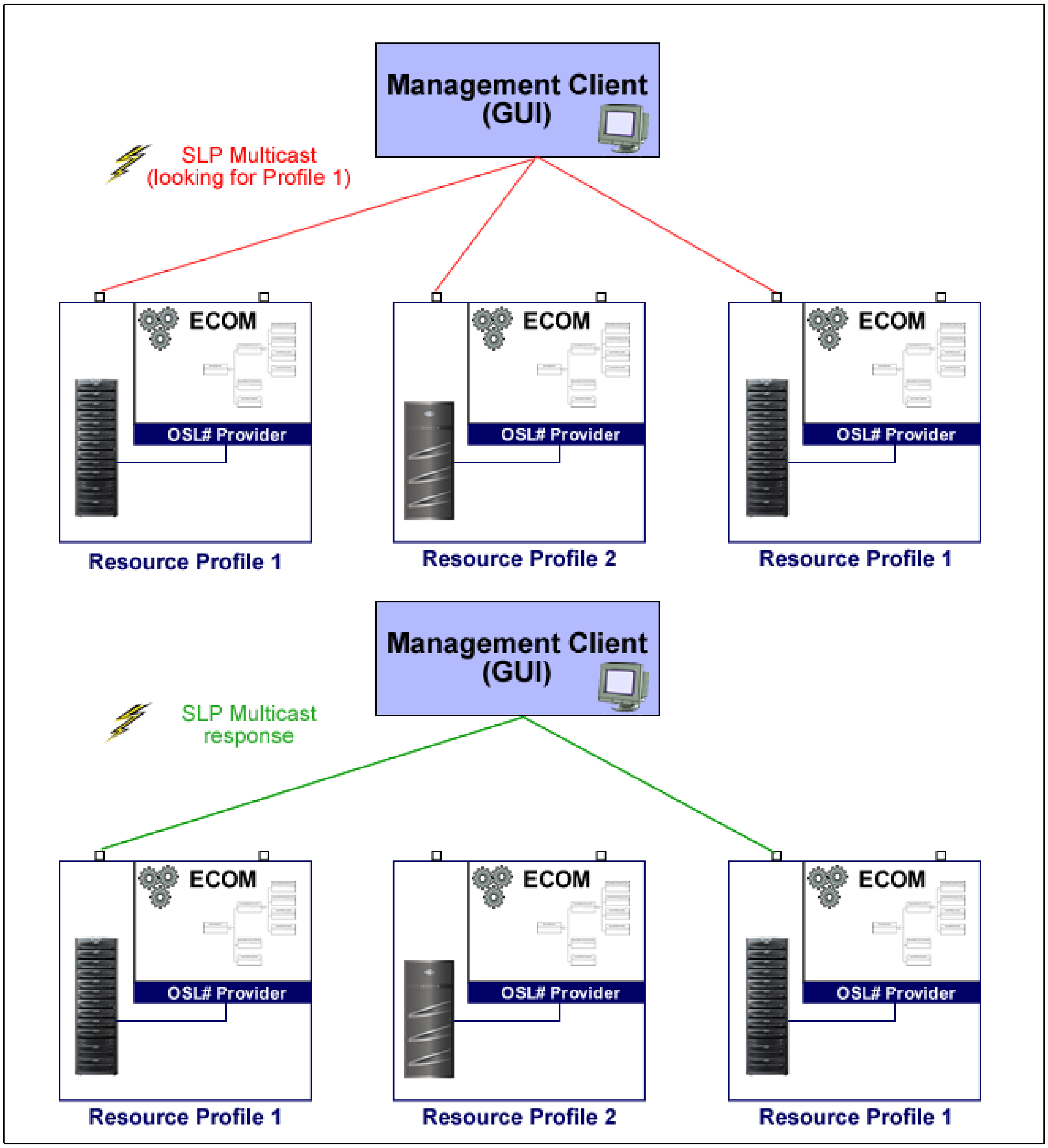

Figure 3 **SLP multicast is used to find needed ECOM instances in a network. Only ECOMs that have the appropriate profile respond, returning information to facilitate direct CIM-XML access.**

CIM intrinsic and extrinsic operations

Once a service consumer and a service provider have established contact, CIM-XML messages are passed between them to transport information about modeled resources and to execute commands (also known as *methods* or *operations*). The types of operations that clients can perform are divided into two categories:

- *Intrinsic operations* are those defined in The DMTF Specification for CIM Operations Over HTTP. These operations are performed against a namespace and modify or obtain information about related CIM elements (classes, instances, qualifiers, properties, etc.).
- *Extrinsic operations* are methods defined within CIM classes themselves. Extrinsic methods define class-specific operations exposed by the model.

As an example, the combination of intrinsic and extrinsic methods allows service consumers to filter through the object types available on an ECOM instance (EnumerateClasses), obtain instances of a specific class type (EnumerateInstances), identify a particular instance to modify (GetInstance), and execute a method defined for it.

See Table 2 on page 23 for a list of supported CIM intrinsic operations. ECOM also supports CIM extrinsic methods.

```
M-POST /cimom HTTP/1.1
HOST: http://172.23.144.237:5988
Content-Type: application/xml; charset="utf-8"
Content-Length: 563
Man: http://www.dmtf.org/cim/mapping/http/v1.0 ; ns=73
73-CIMOperation: MethodCall
73-CIMMethod: EnumerateInstances
73-CIMObject: root/ecom

<?xml version="1.0" encoding="UTF-8"?>
<CIM DTDVERSION="2.0" CIMVERSION="2.0">
     <MESSAGE ID="010080128" PROTOCOLVERSION="1.0">
         <SIMPLEREQ>
             <IMETHODCALL NAME="EnumerateInstances">
                <LOCALNAMESPACEPATH>
                   <NAMESPACE NAME="root"/>
                   <NAMESPACE NAME="interop"/>
                </LOCALNAMESPACEPATH>
                <IPARAMVALUE NAME="ClassName">
                   <CLASSNAME NAME="CIM_System"/>
                </IPARAMVALUE>
                <IPARAMVALUE NAME="DeepInheritance">
                   <VALUE>FALSE</VALUE>
                </IPARAMVALUE>
                <IPARAMVALUE NAME="IncludeClassOrigin">
                   <VALUE>FALSE</VALUE>
                </IPARAMVALUE>
             </IMETHODCALL>
         </SIMPLEREQ>
     </MESSAGE>
</CIM>
```

Figure 4 An example of the EnumerateInstances CIM intrinsic method in a CIM-XML message

SMI-S Server (CIMOM) profile

In addition to querying information about the resources exposed through a CIMOM such as ECOM, SMI-S requires a *server profile* that allows clients to query information about ECOM itself, including the profiles it supports.

CQL

The CIM Query Language (CQL) is used to filter through CIM elements and identify those that meet the requirements of the query. CQL queries help clients filter through the potentially large number of classes and instances that could be exposed through ECOM without having to issue several CIM operations, such as `EnumerateInstances`, and iterate through the data sets returned.

Additional information for modeling

For additional information about the technologies discussed here, consult the following resources:

- DMTF web-based tutorial, covering CIM, WBEM, and profiles
  http://www.wbemsolutions.com/tutorials/DMTF/initiatives.ht
  ml
- DMTF Common Information Model (CIM) web site
  http://www.dmtf.org/standards/cim/
- SNIA Storage Management Initiative (SMI) web site
  http://www.snia.org/smi/home/
- Overview of the EMC Common Information Model (ECIM)
  https://corpusweb172.corp.emc.com/eRoom/spoadvtech/EMC
  SMI/0_135a

ECOM supported standards matrix

ECOM supports the following standards:

Table 1 Standards supported by ECOM

Standard	Compliance	Standard Location
SMI-S 1.2	Full	http://www.snia.org/publicreview
CIM schema version 2.14	Full	http://www.dmtf.org/standards/cim/cim_schema_v214/
CIM-XML version 2.2	Full	http://www.dmtf.org/standards/wbem/DSP201.html
CIM- Operations over HTTP version 1.2	Full	http://www.dmtf.org/standards/published_documents/DSP200.pdf
Service Location Protocol (SLP) version 2	Full	http://tools.ietf.org/html/rfc2608
SSL 3.0 / TLS 1.0	Full	SSL 3.0 http://wp.netscape.com/eng/ssl3/ TLS 1.0 http://www.ietf.org/rfc/rfc2246.txt
CIM Query Language (CQL)	Partial. Only Basic Query and Embedded Properties are supported.	http://www.dmtf.org/standards/documents/WBEM/DSP0202.pdf
HTTP 1.0/1.1	Full	HTTP 1.0 http://www.w3.org/Protocols/rfc1945/rfc1945 HTTP 1.1 http://www.ietf.org/rfc/rfc2616.txt

ECOM supports the following CIM profiles and intrinsic operations:

Table 2 **Supported CIM profiles for intrinsic operations**[a]

Profile	Supported intrinsic operation
Basic Read	GetClass EnumerateClasses EnumerateClassNames ExecQuery GetInstance EnumerateInstances EnumerateInstanceNames GetProperty
Instance Modification	CreateInstance DeleteInstance ModifyInstance
AssociationTraversal	Associators AssociatorNames References ReferenceNames
Basic Write	Set Property

a.ECOM also supports CIM extrinsic methods (InvokeMethod)

The following CIM intrinsics are NOT currently supported but may be included in future releases of ECOM:

- DeleteClass
- CreateClass
- ModifyClass
- GetQualifier
- SetQualifier
- DeleteQualifier
- EnumerateQualifiers

WS-Management support

This section maps CIM functions to their WS-MAN equivalents.

GetInstance()

The `wsa:Action` URI is:
http://schemas.xmlsoap.org/ws/2004/09/transfer/Get

The following example shows a WS Management envelope that issues an `GetInstance() request`.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/wsman.xsd">
      <s:Header>
            <a:To>http://10.5.222.240:5988</a:To>
            <w:ResourceURI
s:mustUnderstand="true">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ro
ot/emc/SP_StorageVolume</w:ResourceURI>
            <a:ReplyTo>
                  <a:Address
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08/addressing/rol
e/anonymous</a:Address>
            </a:ReplyTo>
            <a:Action
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/09/transfer/Get</
a:Action>
<a:MessageID>uuid:109CE460-C546-4045-99A6-DF9A4F2B5F4A</a:MessageID>
            <w:SelectorSet>
                  <w:Selector Name="__cimnamespace">root/emc</w:Selector>
                  <w:Selector
Name="CreationClassName">SP_StorageVolume</w:Selector>
                  <w:Selector Name="DeviceID">Volume 1</w:Selector>
                  <w:Selector
Name="SystemCreationClassName">SP_StorageSystem</w:Selector>
                  <w:Selector Name="SystemName">Big Array</w:Selector>
            </w:SelectorSet>
      </s:Header>
      <s:Body/>
</s:Envelope>
```

Figure 5 WS Management request - GetInstance()

The following shows an example ECOM response to a WS Management `GetInstance()` request.

```
<?xml version="1.0"?>
<s:Envelope xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">
  <s:Header>
    <a:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:To>
    <a:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse</a:Action>
    <a:RelatesTo>uuid:109CE460-C546-4045-99A6-DF9A4F2B5F4A</a:RelatesTo>
    <a:MessageID>uuid:89cd6a6b-ef1d-4df8-a065-43b6ed04</a:MessageID>
  </s:Header>
  <s:Body>
    <i:SP_StorageVolume
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      <i:CreationClassName>SP_StorageVolume</i:CreationClassName>
      <i:DeviceID>Volume 1</i:DeviceID>
      <i:SystemCreationClassName>SP_StorageSystem</i:SystemCreationClassName>
      <i:SystemName>Big Array</i:SystemName>
      <i:BlockSize>512</i:BlockSize>
      <i:ConsumableBlocks>128</i:ConsumableBlocks>
      <i:ExtentStatus>15</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
      <i:NumberOfBlocks>42</i:NumberOfBlocks>
      <i:OperationalStatus>2</i:OperationalStatus>
      <i:OperationalStatus>17</i:OperationalStatus>
    </i:SP_StorageVolume>
  </s:Body>
</s:Envelope>
```

Figure 6 ECOM response to WS Management GetInstance() request

DeleteInstance()

The `wsa:Action` uri is `http://schemas.xmlsoap.org/ws/2004/09/transfer/Delete`.

The following example shows a WS Management envelope that issues a `DeleteInstance()` request.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/wsman.xsd">
      <s:Header>
            <a:To>http://localhost:5985/wsman</a:To>
            <w:ResourceURI
s:mustUnderstand="true">http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/EM
C_ListenerDestinationWSManagement</w:ResourceURI>
            <a:ReplyTo>
                  <a:Address
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08/addressing/rol
e/anonymous</a:Address>
            </a:ReplyTo>
            <a:Action
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/09/transfer/Delet
e</a:Action>
            <w:MaxEnvelopeSize
s:mustUnderstand="true">153600</w:MaxEnvelopeSize>

<a:MessageID>uuid:57178E4E-A018-4DB8-A5D5-E1121189502C</a:MessageID>
            <w:Locale xml:lang="en-US" s:mustUnderstand="false" />
            <p:DataLocale xml:lang="en-US" s:mustUnderstand="false" />
            <w:SelectorSet>
                  <w:Selector Name="__cimnamespace">interop</w:Selector>
                  <w:Selector
Name="CreationClassName">EMC_ListenerDestinationWSManagement</w:Selector>
                  <w:Selector
Name="Name">uuid:2fa47704-b2b3-4681-b7f4-e6b1ea95</w:Selector>
                  <w:Selector
Name="SystemCreationClassName">ECOM_System</w:Selector>
                  <w:Selector
Name="SystemName">emc201.tpa-eld.localdomain</w:Selector>
            </w:SelectorSet>
      </s:Header>
      <s:Body/>
</s:Envelope>
```

Figure 7 WS Management request - DeleteInstance()

The following shows an example ECOM response to a WS Management `DeleteInstance()` request. Notice that this is an empty envelope.

```
<?xml version="1.0"?>

<s:Envelope xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">

  <s:Header>
    <a:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</a:To>
    <a:Action>http://schemas.xmlsoap.org/ws/2004/09/transfer/DeleteResponse</a:Action>
    <a:RelatesTo>uuid:57178E4E-A018-4DB8-A5D5-E1121189502C</a:RelatesTo>
    <a:MessageID>uuid:a550e8d6-f6f6-4b40-8fb2-79ed81d2</a:MessageID>
  </s:Header>
  <s:Body/>
</s:Envelope>
```

Figure 8 ECOM response to WS Management DeleteInstance() request

Associators()

The `Associators` CIM operation is mapped in WS-Management through an `EnumerateInstances` operation.

- The `wsa:Action` uri is `http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse`
- `FilterDialect` is `AssociationFilter`

The following packets show a complete sequence of message exchanges between a client and ECOM for Associators.

Client Request

The Associators operation is implemented using EnumerateInstances operation as its base, with a specific dialect filter, `http://schemas.dmtf.org/wbem/wsman/1/cimbinding/associationFilter`. As parameters for this filter the client must pass the EPR of the object on which we want to execute the Associators operation.

The client requests an `EnumerateObjectAndEPR` response. We could send `EnumerateEPR`, or nothing (which would mean an enumerate objects). There is no limitation on the value of the tag `wsman:EnumerationMode.` If the client is interested only in the resulting instances' key properties, the client can use `wsman:EnumerationMode` with value `EnumerateEPR`.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer">
  <env:Header>
    <wsa:To>http://localhost:5988/wsman</wsa:To>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate</wsa:Action>
    <wsa:ReplyTo>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Addr
ess>
    </wsa:ReplyTo>
    <wsa:MessageID>uuid:925a6b87-f4fa-4914-9f29-7811f665d183</wsa:MessageID>
    <wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/*</wsman:ResourceURI>
  </env:Header>
  <env:Body>
    <wsen:Enumerate>
      <wsman:EnumerationMode>EnumerateObjectAndEPR</wsman:EnumerationMode>
      <wsman:Filter
Dialect="http://schemas.dmtf.org/wbem/wsman/1/cimbinding/associationFilter">
        <wsmb:AssociatedInstances
xmlns:wsmb="http://schemas.dmtf.org/wbem/wsman/1/cimbinding.xsd">
          <wsmb:Object>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
</wsa:Address>
            <wsa:ReferenceParameters>

<wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ECOM_System</wsma
n:ResourceURI>
              <wsman:SelectorSet>
                <wsman:Selector Name="__cimnamespace">interop</wsman:Selector>
                <wsman:Selector Name="CreationClassName">ECOM_System</wsman:Selector>
                <wsman:Selector Name="Name">emc201.tpa-eld.localdomain</wsman:Selector>
              </wsman:SelectorSet>
            </wsa:ReferenceParameters>
          </wsmb:Object>
        </wsmb:AssociatedInstances>
      </wsman:Filter>
    </wsen:Enumerate>
  </env:Body>
</env:Envelope>
```

Figure 9 Associators Client Request

ECOM Response - Enumeration Context

ECOM responds with an enumeration context.

```
<?xml version="1.0"?>

<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">

  <s:Header>
    <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse</wsa:A
ction>
    <wsa:RelatesTo>uuid:925a6b87-f4fa-4914-9f29-7811f665d183</wsa:RelatesTo>
    <wsa:MessageID>uuid:b46e2fc7-3bcb-456d-83ad-06571077</wsa:MessageID>
  </s:Header>
  <s:Body>
    <wsen:EnumerateResponse>

<wsen:EnumerationContext>O_zrqig08twjyf928ir3z-----------------_ObjAndEPR_</wsen:Enume
rationContext>
    </wsen:EnumerateResponse>
  </s:Body>
</s:Envelope>
```

Figure 10 ECOM Response - Enumeration Context

Client Sends a Pull Request

Client sends the Pull request, passing the received `EnumerationContext` asking for up to 50 `MaxElements`.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer">

  <env:Header>
    <wsa:To>http://localhost:5988/wsman</wsa:To>
    <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull</wsa:Action>
    <wsa:ReplyTo>

<wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:Addr
ess>
    </wsa:ReplyTo>
    <wsa:MessageID>uuid:d2070275-752d-4f65-b8f7-d5582f252b98</wsa:MessageID>
    <wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/*</wsman:ResourceURI>
  </env:Header>
  <env:Body>
    <wsen:Pull>

<wsen:EnumerationContext>O_zrqig08twjyf928ir3z-----------------_ObjAndEPR_</wsen:Enume
rationContext>
      <wsen:MaxElements>50</wsen:MaxElements>
    </wsen:Pull>
  </env:Body>
</env:Envelope>
```

Figure 11 Client Pull Request

ECOM Response - Instances and EPRs

ECOM responds with the instances and respectively EPR's. The result here is truncated.

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xml:lang="en-US">
  <s:Header>
    <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>
    <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/PullResponse</wsa:Action>
    <wsa:RelatesTo>uuid:d2070275-752d-4f65-b8f7-d5582f252b98</wsa:RelatesTo>
    <wsa:MessageID>uuid:ca941195-6791-4f28-8eb7-877b2571</wsa:MessageID>
  </s:Header>
  <s:Body>
    <wsen:PullResponse>
      <wsen:EnumerationContext/>
      <wsen:Items>
        <wsen:Item>
          <i:ECOM_CIMXMLCommunicationMechanism
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ECOM_CIMXMLCommunicationMechanism"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <i:CIMXMLProtocolVersion>1</i:CIMXMLProtocolVersion>
            <i:CIMValidated>false</i:CIMValidated>
            <i:CommunicationMechanism>2</i:CommunicationMechanism>
<i:OtherCommunicationMechanismDescription>CIM-XML</i:OtherCommunicationMechanismDescription>
            <i:FunctionalProfilesSupported>2</i:FunctionalProfilesSupported>
            <i:FunctionalProfilesSupported>3</i:FunctionalProfilesSupported>
            <i:FunctionalProfilesSupported>5</i:FunctionalProfilesSupported>
            <i:FunctionalProfilesSupported>6</i:FunctionalProfilesSupported>
            <i:FunctionalProfilesSupported>9</i:FunctionalProfilesSupported>
            <i:FunctionalProfileDescriptions>Basic Read</i:FunctionalProfileDescriptions>
            <i:FunctionalProfileDescriptions>Basic Write</i:FunctionalProfileDescriptions>
            <i:FunctionalProfileDescriptions>Instance Manipulation</i:FunctionalProfileDescriptions>
            <i:FunctionalProfileDescriptions>Association Traversal</i:FunctionalProfileDescriptions>
            <i:FunctionalProfileDescriptions>Indications</i:FunctionalProfileDescriptions>
            <i:MultipleOperationsSupported>true</i:MultipleOperationsSupported>
            <i:AuthenticationMechanismsSupported>3</i:AuthenticationMechanismsSupported>
            <i:AuthenticationMechanismDescriptions>Basic</i:AuthenticationMechanismDescriptions>
            <i:Version>1.0</i:Version>
            <i:AdvertiseTypes>3</i:AdvertiseTypes>
            <i:SystemCreationClassName>ECOM_System</i:SystemCreationClassName>
            <i:SystemName>emc201.tpa-eld.localdomain</i:SystemName>
            <i:CreationClassName>ECOM_CIMXMLCommunicationMechanism</i:CreationClassName>
            <i:EnabledState>5</i:EnabledState> <i:OtherEnabledState/>
            <i:RequestedState>12</i:RequestedState>
            <i:EnabledDefault>2</i:EnabledDefault>
            <i:OperationalStatus>2</i:OperationalStatus>
            <i:StatusDescriptions>OK</i:StatusDescriptions>
            <i:Name>slp:http://emc201.tpa-eld.localdomain:5988</i:Name>
            <i:Caption/><i:Description/>
            <i:ElementName>cim-xml Adapter</i:ElementName>
          </i:ECOM_CIMXLCommunicationMechanism>
          <wsa:EndpointReference>
            <wsa:Address>http://emc201.tpa-eld.localdomain:5985</wsa:Address>
            <wsa:ReferenceParameters>
<wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/ECOM_CIMXMLCommunicationMechanism</wsman:Resou
rceURI>
              <wsman:SelectorSet>
                <wsman:Selector Name="__cimnamespace">interop</wsman:Selector>
                <wsman:Selector Name="SystemCreationClassName">ECOM_System</wsman:Selector>
                <wsman:Selector Name="CreationClassName">ECOM_CIMXMLCommunicationMechanism</wsman:Selector>
                <wsman:Selector Name="Name">slp:http://emc201.tpa-eld.localdomain:5988</wsman:Selector>
                <wsman:Selector Name="SystemName">emc201.tpa-eld.localdomain</wsman:Selector>
              </wsman:SelectorSet>
            </wsa:ReferenceParameters>
          </wsa:EndpointReference>
        </wsen:Item>
...
        </wsen:Items><wsen:EndOfSequence/></wsen:PullResponse></s:Body>
</s:Envelope>
```

Figure 12 ECOM - Instances and EPRs

OpenAssociatorInstances()

- The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
- `EnumerationMode` is `NULL` or `EnumerateObjectAndEPR`
- `FilterDialect` is `AssociatedInstances`

PullInstancesWithPath()

- The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull
- `Enumeration Context` is `OBJECT`

PullInstancePaths()

- The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/enumeration/Pull
- `Enumeration Context` is `INSTANCENAME`

CloseEnumerationContext()

The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/enumeration/Release

ModifyInstance()

The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/transfer/Put

CreateInstance()

The `wsa:Action` uri is http://schemas.xmlsoap.org/ws/2004/09/transfer/Create

ExecQuery()

- The `wsa:Action` URI is: http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
- `wsman:Filter Dialect` is http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf .
- The body of the request must include this element.

  `<wsen:OptimizeEnumeration/>`

The following example shows a WS Management envelope that issues an `ExecQuery() request`.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer">

  <s:Header>
      <wsa:ReplyTo>
            <wsa:Address
s:mustUnderstand="true">http://schemas.xmlsoap.org/ws/2004/08/addressing/rol
e/anonymous</wsa:Address>
      </wsa:ReplyTo>
    <wsa:MessageID>uuid:7b353aa1-9c81-4fac-9a73-08d54fa61e3d</wsa:MessageID>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate</wsa
:Action>

<wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/*</wsman:ResourceURI
>
    <wsman:SelectorSet>
      <wsman:Selector Name="__cimnamespace">root/emc</wsman:Selector>
    </wsman:SelectorSet>
  </s:Header>
  <s:Body>

    <wsen:Enumerate>
      <wsen:OptimizeEnumeration />
      <wsman:Filter
Dialect="http://schemas.dmtf.org/wbem/cql/1/dsp0202.pdf">SELECT A.BlockSize,
A.NumberOfBlocks FROM SP_StorageVolume AS A</wsman:Filter>
    </wsen:Enumerate>
  </s:Body>
</s:Envelope>
```

Figure 13 WS Management request - ExecQuery()

The following shows an example ECOM response to a WS Management `ExecQuery()` request.

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">

  <s:Header>
    <wsa:To>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</wsa:To>

<wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse</wsa:A
ction>
    <wsa:RelatesTo>uuid:7b353aa1-9c81-4fac-9a73-08d54fa61e3d</wsa:RelatesTo>
    <wsa:MessageID>uuid:ec0b9029-2d75-44ed-a2c8-992ad3f3</wsa:MessageID>
  </s:Header>
  <s:Body>
    <wsen:PullResponse>
      <wsen:EnumerationContext/>
      <wsen:Items>

<wsman:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <i:BlockSize>512</i:BlockSize>
          <i:NumberOfBlocks>42</i:NumberOfBlocks>
        </wsman:XmlFragment>

        <wsman:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <i:BlockSize>512</i:BlockSize>
          <i:NumberOfBlocks>42000</i:NumberOfBlocks>
        </wsman:XmlFragment>

        <wsman:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <i:BlockSize>512</i:BlockSize>
          <i:NumberOfBlocks>12345678</i:NumberOfBlocks>
        </wsman:XmlFragment>

        <wsman:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
          <i:BlockSize>512</i:BlockSize>
          <i:NumberOfBlocks>3</i:NumberOfBlocks>
        </wsman:XmlFragment>
</wsen:Items>
      <wsen:EndOfSequence/>
    </wsen:PullResponse>
  </s:Body>
</s:Envelope>
```

Figure 14 ECOM response to WS Management ExecQuery() request

Limitations of ECOM CIM Operations

Table 3 on page 36 lists known limitations of ECOM's CIM operations and their correspondence to its WS-Management support.

Table 3 CIM operation support and WS-Management limitations

CIM Intrinsic Operation	WS-Management support?	Supported Parameters	Unsupported parameters
Associators()	Yes	ObjectName=InstanceNames, AssociationClass, ResultClass, Role, ResultRole	ObjectName=ClassName, IncludeClassOrigin & PropertyList
AssociatorNames()	Yes	ObjectName=InstanceNames, AssociationClass, ResultClass, Role, ResultRole	ObjectName=ClassName
CreateInstance()	Yes	NewInstance	
DeleteInstance()	Yes	InstanceName	
EnumerateInstanceNames()	Yes	ClassName	
EnumerateInstances()	Yes	ClassName	
GetClass ()	No (no mapping in DSP0207)		
GetInstance()	Yes	InstanceName	IncludeClassOrigin, PropertyList
ModifyInstance()	Yes	ModifiedInstance	PropertyList
ReferenceNames()	Yes	ObjectName=InstanceName, ResultClass, Role	ObjectName=ClassName
References()	Yes	ObjectName=InstanceName, ResultClass, Role	ObjectName=ClassName, IncludeClassOrigin & PropertyList
OpenEnumerateInstancePath s	Yes	ClassName	FilterQueryLanguage, FilterQuery, OperationTimeout, ContinueOnError
OpenEnumerateInstances	Yes	ClassName	DeepInheritance, IncludeClassOrigina, PropertyList, FilterQueryLanguage, FilterQuery, OperationTimeout, ContinueOnError

Table 3 CIM operation support and WS-Management limitations

CIM Intrinsic Operation	WS-Management support?	Supported Parameters	Unsupported parameters
OpenAssociatorInstances	Yes	EnumerationContext, InstanceName, EndOfSequence, AssociationClass, ResultRole, Role, ResultRole, MaxObjectCount	IncludeClassOrigina, PropertyList, FilterQueryLanguage, FilterQuery, OperationTimeout, ContinueOnError
OpenAssociatorPaths	Yes	EnumerationContext, InstanceName, EndOfSequence, AssociationClass, ResultRole, Role, ResultRole, MaxObjectCount	FilterQueryLanguage, FilterQuery, OperationTimeout, ContinueOnError
PullInstancePaths	Yes	EnumerationContext, EndOfSequence, MaxObjectCount	
PullInstancesWithPath	Yes	EnumerationContext, EndOfSequence, MaxObjectCount	
CloseEnumeration	Yes	EnumerationContext	
ExecQuery	Yes	Query, QueryLanguage	

Optional features in WS-Management, such as OptimizeEnumeration, are not supported unless specifically stated.

The following intrinsic methods in WS-Management can only take instance names as input parameters:

- Associators
- AssociatorNames
- References
- ReferenceNames

<wsen:Expires> element support

ECOM now supports the <wsen:Expires> element, allowing ECOM clients to increase system security and specify the allowed duration for enumeration requests. <wsen:Expires> maps directly to the CIM operation timeout as an enumeration context timeout. Figure Figure 15 on page 38 provides an example.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<env:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
     xmlns:env="http://www.w3.org/2003/05/soap-envelope"
     xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
     xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
     xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
     xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer">
    <env:Header>
       <wsa:To>http://localhost:5985/wsman</wsa:To>
       <wsa:Action>http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumera
         t</wsa:Action>
       <wsa:ReplyTo>
       <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/an
         onymous</wsa:Address>
       </wsa:ReplyTo>
       <wsa:MessageID>uuid:7150fae3-0c6e-407e-a4a1-7369748abbc0</wsa:Message
         ID>
       <wsman:ResourceURI>http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/
         CIM_ManagedElement</wsman:ResourceURI>
       <wsman:MaxEnvelopeSize
         s:mustUnderstand="true">150000</wsman:MaxEnvelopeSize>
       <wsman:Locale xml:lang="en-US" s:mustUnderstand="false" />
       <wsman:SelectorSet>
       <wsman:Selector Name="__cimnamespace">interop</wsman:Selector>
       </wsman:SelectorSet>
    </env:Header>
    <env:Body>
         <wsen:Enumerate>
         <wsman:EnumerationMode>EnumerateEPR</wsman:EnumerationMode>
         <wsen:Expires>2012-03-01T10:01:01</wsen:Expires>
         </wsen:Enumerate>
    </env:Body>
</env:Envelope>
```

Figure 15 <wsen:Expires /> example

Note: The enumeration operation timeout value can also be set globally for ECOM through the `DefaultPulledEnumOperationTimeout` parameter in ECOM_settings.xml.

The <wsen:Expires> element can hold a time value of type xs:dateTime or xs:duration.

```
  <s:Body …>
    <wsen:Enumerate …>
      …
      <wsen:Expires>[xs:dateTime |
xs:duration]</wsen:Expires> ?
      …
    </wsen:Enumerate>
  </s:Body>
```

dateTime is specified in the form "YYYY-MM-DDThh:mm:ss" where:

- YYYY indicates the year
- MM indicates the month
- DD indicates the day
- T indicates the start of the required time section
- hh indicates the hour
- mm indicates the minute
- ss indicates the second

All components of this format are required for use with ECOM.

If a value of type xs:dateTime is passed, ECOM calculates how many seconds are left between the moment it received the message and the specified dateTime. All calculations are based on ECOM's current timezone.

An example of an xs:duration value of 30 seconds is:

```
  …
  <wsen:Expires>PT30S</wsen:Expires>
  …
```

Another example, which will expire in 1 year, 2 months and 3 days, is:

```
  …
  <wsen:Expires>P1Y2M3D</wsen:Expires>
  …
```

The duration for xs:duration is specified in the form "PnYnMnDTnHnMnS" where:

- P indicates the period (required)
- Y indicates the number of years
- M indicates the number of months
- D indicates the number of days
- T indicates the start of a time section (required if you are going to specify hours, minutes, or seconds)
- H indicates the number of hours
- M indicates the number of minutes
- S indicates the number of seconds

Extrinsic Methods

ECOM supports extrinsic method invocation through WS-Management. Figure 16 on page 42 demonstrates an example request.

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
    xmlns:n1="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/QE_QETestMeth
    od" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<s:Header>
   <wsa:Action
      s:mustUnderstand="true">http://schemas.dmtf.org/wbem/wscim/1/cim-schem
      a/2/QE_QETestMethods/CompareStaticData</wsa:Action>
   <wsa:To s:mustUnderstand="true">http://9.123.251.94:5985/wsman</wsa:To>
   <wsman:ResourceURI
      s:mustUnderstand="true">http://schemas.dmtf.org/wbem/wscim/1/cim-schem
      a/2/QE_QETestMethods</wsman:ResourceURI>
   <wsa:MessageID
      s:mustUnderstand="true">uuid:a9d78e51-5943-1943-8002-7ec0ed251100</wsa
      :MessageID>
   <wsa:ReplyTo>
      <wsa:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/ano
        nymous</wsa:Address>
   </wsa:ReplyTo>
   <wsman:SelectorSet>
      <wsman:Selector Name="__cimnamespace">root/emc</wsman:Selector>
      <wsman:Selector Name="name">TestMethodsInstanceName</wsman:Selector>
   </wsman:SelectorSet>
</s:Header>
<s:Body>
  <n1:RequestStateChange_INPUT>
      <n1:inData
         xmlns:e="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/QE_QETest
         Data" xsi:type="e:QE_QETestData_Type">
         <e:name>TestDataInstanceName</e:name>
         <e:IntData>42</e:IntData>
         <e:IntArrayData>17</e:IntArrayData>
         <e:IntArrayData>18</e:IntArrayData>
         <e:IntArrayData>12</e:IntArrayData>
         <e:IntArrayData>33</e:IntArrayData>
         <e:StringArrayData>What</e:StringArrayData>
         <e:StringArrayData>is</e:StringArrayData>
         <e:StringArrayData>six</e:StringArrayData>
         <e:StringArrayData>times</e:StringArrayData>
         <e:StringArrayData>nine</e:StringArrayData>
         <e:StringArrayData>?</e:StringArrayData>
         <e:StringData>Hello cruel world?</e:StringData>
      </n1:inData>
  </n1:RequestStateChange_INPUT>
</s:Body>
</s:Envelope>
```

Figure 16 WS-Management Extrinsic method invocation example with an embedded instance

XML Fragment Support

The WS-Management specification defines the concept of fragment-level (property) access of resources. This fragment-level access applies to the operations get, put, delete, create and enumeration but we are only going to describe here the fragment level operations get and enumeration, because that are the ones currently implemented.

For the operations get, put, delete and create, the fragment level selection is done through the SOAP Header tag

```
<wsman:FragmentTransfer s:mustUnderstand="true">
   xpath expression
</wsman:FragmentTransfer>
```

The expression used is a subset of the XPath 1.0 specification. For a Get, the XPath expression is able to select a node (and its child or children), and a specific property value of node. For an enumerate, the XPath expression filters out instances from the result set and, from the filtered instances, selects only a subset of the oroperties of those instances.

Fragment Level Get Operations

The fragment level Get operation is very similar to the regular Get operation, but ECOM adds a `wsman:FragmentTransfer` tag at the SOAP header. For example, the following request selects a specific `SP_StorageVolume` instance as the regular Get does, but also limits the output from ECOM to only 3 properties of this instance: `SystemName`, `BlockSize` and `DeviceID`.

Table 4 XML Fragment Example

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
   xmlns:p="http://schemas.microsoft.com/wbem/wsman/1/wsman.xsd">
   <s:Header>
      <a:To>http://10.5.222.240:5988</a:To>
      <w:ResourceURI
      s:mustUnderstand="true">
http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/root/emc/SP_StorageVolume
      </w:ResourceURI>
      <a:ReplyTo>
         <a:Address s:mustUnderstand="true">
http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
         </a:Address>
      </a:ReplyTo>
```

Table 4 XML Fragment Example

```
        <a:Action s:mustUnderstand="true">
            http://schemas.xmlsoap.org/ws/2004/09/transfer/Get
        </a:Action>
        <w:MaxEnvelopeSize s:mustUnderstand="true">
            153600
        </w:MaxEnvelopeSize>
        <a:MessageID>
            uuid:109CE460-C546-4045-99A6-DF9A4F2B5F4A
        </a:MessageID>
        <w:Locale xml:lang="en-US" s:mustUnderstand="false" />
        <p:DataLocale xml:lang="en-US" s:mustUnderstand="false" />
        <w:SelectorSet>
            <w:Selector Name="__cimnamespace">root/emc</w:Selector>
            <w:Selector Name="CreationClassName">SP_StorageVolume</w:Selector>
            <w:Selector Name="DeviceID">Volume 1</w:Selector>
            <w:Selector Name="SystemCreationClassName">
                SP_StorageSystem
            </w:Selector>
            <w:Selector Name="SystemName">
                Big Array
            </w:Selector>
        </w:SelectorSet>
        <w:FragmentTransfer s:mustUnderstand="true">
            //*[self::i:SystemName or self::i:BlockSize or self::i:DeviceID]
        </w:FragmentTransfer>
    </s:Header>
    <s:Body/>
</s:Envelope>
```

The ECOM response to the above request is shown in Table 5.

Table 5 ECOM Response to Example in Table 5

```
<?xml version="1.0"?>
<s:Envelope xmlns:a="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:w="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">

   <s:Header>
      <a:To>
         http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
      </a:To>

   <a:Action>
      http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse
   </a:Action>
   <a:RelatesTo>
      uuid:109CE460-C546-4045-99A6-DF9A4F2B5F4A
   </a:RelatesTo>
   <a:MessageID>
      uuid:dd482640-54c6-4783-874f-75586961
   </a:MessageID>
   </s:Header>

   <s:Body>
      <w:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
         <i:DeviceID>Volume 1</i:DeviceID>
         <i:SystemName>Big Array</i:SystemName>
         <i:BlockSize>512</i:BlockSize>
      </w:XmlFragment>
   </s:Body>
</s:Envelope>
```

XPath examples for Get operation

Consider the following instance of the `SP_StorageVolume` class:

Table 6 SP_StorageVolume Class Example

```
...
<s:Body>
<i:SP_StorageVolume
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <i:CreationClassName>SP_StorageVolume</i:CreationClassName>
   <i:DeviceID>Volume 1</i:DeviceID>
   <i:SystemCreationClassName>SP_StorageSystem</i:SystemCreationClassName>
   <i:SystemName>Big Array</i:SystemName>
   <i:BlockSize>512</i:BlockSize>
   <i:ConsumableBlocks>128</i:ConsumableBlocks>
   <i:ExtentStatus>15</i:ExtentStatus>
   <i:ExtentStatus>6</i:ExtentStatus>
   <i:NumberOfBlocks>42</i:NumberOfBlocks>
   <i:OperationalStatus>2</i:OperationalStatus>
   <i:OperationalStatus>17</i:OperationalStatus>
</i:SP_StorageVolume>
</s:Body>
...
```

The following examples show ECOM's response to a variety of XPath expressions:

Table 7 ECOM Response / XPath Expression matrix

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   //*[name()='i:SystemName' or name()='i:BlockSize' or`
	`          name()='i:DeviceID']`
	`</w:FragmentTransfer>`

ECOM Response	`<w:XmlFragment`
	`xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_`
	`StorageVolume">`
	`<i:DeviceID>Volume 1</i:DeviceID>`
	`   <i:SystemName>Big Array</i:SystemName>`
	`   <i:BlockSize>512</i:BlockSize>`
	`</w:XmlFragment>`

Comment	This xpath has the same effect as
	`// *[self::i:SystemName or self::i:BlockSize or`
	`      self::i:DeviceID]`

Table 7 ECOM Response / XPath Expression matrix

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   /i:SP_StorageVolume/i:ConsumableBlocks`
	`</w:FragmentTransfer>`

ECOM Response	`<w:XmlFragment`
	`xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_`
	`StorageVolume">`
	`   <i:ConsumableBlocks>128</i:ConsumableBlocks>`
	`</w:XmlFragment>`

Comment	

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   /i:SP_StorageVolume/i:ConsumableBlocks/text()`
	`</w:FragmentTransfer>`

ECOM Response	`<s:Body>`
	`   <w:XmlFragment>128</w:XmlFragment>`
	`</s:Body>`

Comment	Note how when we append the 'text()' xpath function to the expression the returned value is not enclosed with the property name tag.

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   count(/i:SP_StorageVolume/*)`
	`</w:FragmentTransfer>`

ECOM Response	`<s:Body>`
	`   <w:XmlFragment>11</w:XmlFragment>`
	`</s:Body>`

Comment	This counts the number of child tags under i:SP_StorageVolume. Note, this is not a properties count, because we can have properties that are arrays and so the array name appers more than once. Like for example the 'ExtentStatus' tag.

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`<!-- remember that name() returns 'i:ExtentStatus' and xpath`
	`index count starts with 1 -->`
	`   count(//*[substring(name(),3,6)='Extent'])`
	`</w:FragmentTransfer>`

ECOM Response	`<s:Body>`
	`   <w:XmlFragment>2</w:XmlFragment>`
	`</s:Body>`

Comment	This counts how many elements we have that has the string 'Extent' starting at position 3 from the return of function name() (which returns the name of the element being evaluated).

Table 7 ECOM Response / XPath Expression matrix

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   count(/i:SP_StorageVolume/descendant::i:OperationalStatus)`
	`</w:FragmentTransfer>`

ECOM Response	`<s:Body>`
	`   <w:XmlFragment>2</w:XmlFragment>`
	`</s:Body>`

Comment	Another form to count the number of properties with a specific name. Here we are counting the number of times `i:OperationalStatus` appears.

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`    //i:ExtentStatus[2]/text()`
	`</w:FragmentTransfer>`

ECOM Response	`<s:Body>`
	`   <w:XmlFragment>6</w:XmlFragment>`
	`</s:Body>`

Comment	Select the second element with the name `i:ExtentStatus`. This XPath expression requests the second element of the `ExtentStatus` array.

XPath Expression	`<w:FragmentTransfer s:mustUnderstand="true">`
	`   //*[text() = 'SP_StorageSystem']`
	`</w:FragmentTransfer>`

ECOM Response	`<w:XmlFragment`
	`xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_`
	`StorageVolume">`
	`   <i:SystemCreationClassName>`
	`      SP_StorageSystem`
	`   </i:SystemCreationClassName>`
	`</w:XmlFragment>`

Comment	Select the node element containing the string `SP_StorageSystem`.

Table 7 ECOM Response / XPath Expression matrix

XPath Expression

```
<w:FragmentTransfer s:mustUnderstand="true">
   concat('The System Name is ',
       "'",//i:SystemName/text(),
       "'",' and it has block size of ',
       //i:BlockSize/text(),
       " and '", //i:NumberOfBlocks/text(),
       "' number of blocks, totalling ",
       //i:BlockSize/text() * //i:NumberOfBlocks/text())
</w:FragmentTransfer>
```

ECOM Response

```
<s:Body>
   <w:XmlFragment>The System Name is 'Big Array' and it has
       block size of 512 and '42' number of blocks, totalling
       21504
   </w:XmlFragment>
</s:Body>
```

Comment

This expression concatenates some property values and also a calculation.

Fragment level Enumeration operation

You can use XMLFragments with enumerations to do the following:

- Select specific properties from all elements. This is effectively a `Get` operation applied to each element of the enumeration.
- Filter out elements in the enumeration based on the instance properties.

For example, to request all elements of an enumeration that have a populated `ExtentStatus` property, send the following enumeration request to ECOM:

Table 8 Request for Enumeration Elements with Specific Property Data

```
<env:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:env="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
   xmlns:wsman="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
   xmlns:wxf="http://schemas.xmlsoap.org/ws/2004/09/transfer">

   <env:Header>
       <wsa:To>http://localhost:5985/wsman</wsa:To>
           <wsa:Action>
               http://schemas.xmlsoap.org/ws/2004/09/enumeration/Enumerate
           </wsa:Action>

           <wsa:ReplyTo>
               <wsa:Address>
               http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
               </wsa:Address>
           </wsa:ReplyTo>
           <wsa:MessageID>
               uuid:7150fae3-0c6e-407e-a4a1-7369748abbc0
           </wsa:MessageID>
           <wsman:ResourceURI>
           http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume
           </wsman:ResourceURI>
           <wsman:MaxEnvelopeSize s:mustUnderstand="true">
               150000
           </wsman:MaxEnvelopeSize>
           <wsman:Locale xml:lang="en-US" s:mustUnderstand="false" />
           <wsman:SelectorSet>
               <wsman:Selector Name="__cimnamespace">root/emc</wsman:Selector>
           </wsman:SelectorSet>
   </env:Header>
   <env:Body>
           <wsen:Enumerate>
               <wsman:Filter
                       Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
                   //i:ExtentStatus
               </wsman:Filter>
           </wsen:Enumerate>
   </env:Body>
</env:Envelope>
```

The XPath expression `//i:ExtentStatus`' makes ECOM return all elements in the enumeration whose property name is `i:ExtentStatus`.

ECOM's response is (omitting the intermediate step of the enumeration context sent by ECOM):

Table 9 ECOM Response to Enumeration Elements with Specific Property Data

```
<?xml version="1.0"?>
<s:Envelope xmlns:ns2="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:ns3="http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd"
   xmlns:ns4="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
   xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xml:lang="en-US">


<s:Header>
   <ns2:To>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
   </ns2:To>
      <ns2:Action>
      http://schemas.xmlsoap.org/ws/2004/09/enumeration/PullResponse
   </ns2:Action>
   <ns2:RelatesTo>
      uuid:a9d78e51-5943-1943-8002-7ec0ed251100
   </ns2:RelatesTo>
      <ns2:MessageID>uuid:09e2b64b-1f25-4a4a-989f-11c77371
   </ns2:MessageID>
</s:Header>

<s:Body>
   <ns4:PullResponse>
      <ns4:EnumerationContext/>
      <ns4:Items>
      <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
         <i:ExtentStatus>15</i:ExtentStatus>
         <i:ExtentStatus>6</i:ExtentStatus>
      </ns3:XmlFragment>

   <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
      <i:ExtentStatus>4</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
   </ns3:XmlFragment>
```

Table 9 ECOM Response to Enumeration Elements with Specific Property Data

```
   <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
      <i:ExtentStatus>15</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
   </ns3:XmlFragment>

   <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
      <i:ExtentStatus>4</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
   </ns3:XmlFragment>

   <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
      <i:ExtentStatus>0</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
   </ns3:XmlFragment>

   <ns3:XmlFragment
xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_StorageVolume">
      <i:ExtentStatus>15</i:ExtentStatus>
      <i:ExtentStatus>6</i:ExtentStatus>
   </ns3:XmlFragment>

   </ns4:Items>
      <ns4:EndOfSequence/>
      </ns4:PullResponse>
   </s:Body>
</s:Envelope>
```

Examples: Varying ExtentStatus

The following examples show the ECOM response to a varying number of 'i:ExtentStatus'. ExtentStatus is an array of integers.

Table 10 ECOM Response / XPath Expression matrix

Request

```
<w:Filter
Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116">
   concat('The System Name is ',
   "'",
   //i:SystemName/text(),
   "'",
   ' and it has block size of ',
   //i:BlockSize/text(),
   " and '",
   //i:NumberOfBlocks/text(),
   "' number of blocks, totalizing ",
   //i:BlockSize/text() * //i:NumberOfBlocks/text())
</w:Filter>
```

Response

```
<s:Body>
   <ns4:PullResponse>
       <ns4:EnumerationContext/>
       <ns4:Items>
           <ns3:XmlFragment>The System Name is 'Big Array'
               and it has block size of 512 and '42' number of
               blocks, totalizing 21504
           </ns3:XmlFragment>
           <ns3:XmlFragment>The System Name is 'Big Array'
               and it has block size of 512 and '42000' number
               of blocks, totalizing 21504000
           </ns3:XmlFragment>
           <ns3:XmlFragment>The System Name is 'Big Array'
               and it has block size of 512 and '12345678' number
               of blocks, totalling 2026019840
           </ns3:XmlFragment>
           <ns3:XmlFragment>The System Name is 'Big Array' and
               it has block size of 512 and '3' number of blocks,
               totalizing 1536
           </ns3:XmlFragment>
           <ns3:XmlFragment>The System Name is 'Big Array'
               and it has block size of 512 and '15' number of
               blocks, totalizing 7680
           </ns3:XmlFragment>
           <ns3:XmlFragment>The System Name is 'Small Array'
               and it has block size of 512 and '117' number
               of blocks, totalizing 59904
           </ns3:XmlFragment>
       </ns4:Items>
       <ns4:EndOfSequence/>
   </ns4:PullResponse>
</s:Body>
```

Comments

As in the Get operation, the XPath expression is applied to each element of the enumeration.

Table 10 ECOM Response / XPath Expression matrix

Request	<w:Filter Dialect="http://www.w3.org/TR/1999/REC-xpath-19991116"> //*[substring(name(),3,6)='Extent']/ancestor::*/i:NumberOfBloc ks[text()<117]/ancestor::*/*[self::i:SystemName or self::i:BlockSize or self::i:DeviceID] </w:Filter>

Response	<s:Body>
	   <ns4:PullResponse>
	      <ns4:EnumerationContext/>
	      <ns4:Items>
	      <ns3:XmlFragment
	xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_
	StorageVolume">
	          <i:DeviceID>Volume 1</i:DeviceID>
	          <i:SystemName>Big Array</i:SystemName>
	          <i:BlockSize>512</i:BlockSize>
	      </ns3:XmlFragment>
	      <ns3:XmlFragment
	xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_
	StorageVolume">
	          <i:DeviceID>Volume 4</i:DeviceID>
	          <i:SystemName>Big Array</i:SystemName>
	          <i:BlockSize>512</i:BlockSize>
	      </ns3:XmlFragment>
	      <ns3:XmlFragment
	xmlns:i="http://schemas.dmtf.org/wbem/wscim/1/cim-schema/2/SP_
	StorageVolume">
	          <i:DeviceID>Volume 5</i:DeviceID>
	          <i:SystemName>Big Array</i:SystemName>
	          <i:BlockSize>512</i:BlockSize>
	      </ns3:XmlFragment>
	      </ns4:Items>
	      <ns4:EndOfSequence/>
	   </ns4:PullResponse>
	</s:Body>

Table 10 ECOM Response / XPath Expression matrix

Comments	The XPath expression can be read like this: • select the node that conatins the substring 'Extent' as its name • get its ancestors nodes. (In this case we know it is only one ancestor - the `i:SP_StorageVolume` node) • Select the node name `i:NumberOfBlocks`' with value less than 117. • From that selection, select all ancestors • From that selection, select the child nodes `i:SystemName`, `i:BlockSize` and `i:DeviceID.` The final result are the nodes `i:SystemName`, `i:BlockSize` and `i:DeviceID` from all instances that have `i:NumberOfBlocks` < 117 and contain some property with the substring 'Extent' in its name.

2

Running ECOM

The following sections describe how to o run ECOM.

Starting and stopping ECOM

The following sections discuss how to start and stop ECOM on supported platforms.

Starting ECOM

The following sections present platform-specific instructions for starting ECOM. To ensure SMI-S compliance, you should also start an SLP service agent (**slpd**) before starting ECOM. See Appendix A, "SLP Configuration" for more information about SLP configuration and startup.

Note: ECOM listens on open and secure ports as defined in `Port_settings.xml`. Before starting ECOM, shut down or reconfigure any processes on the ECOM platform that may be listening on the same ports at the same IP address as ECOM. For example, web servers such as Microsoft IIS or Apache may listen at default ports 80 or 443 and cause a conflict with ECOM. Optionally, you can also configure ECOM to listen on different ports. "ECOM port security" on page 104 provides additional information.

Linux

To start ECOM on a Linux-based platform:

1. Open a shell to obtain a command prompt.
2. Navigate to `<ECOM_install_dir>`\bin.
3. Execute the command:

   ```
   ./ecom -c <ecom_conf_dir>
   ```

   where `<ecom_conf_dir>` is the full path to the directory containing the file `ECOM_settings.xml`. ECOM will start in the command window.

Note: You can also set the path for `<ecom_conf_dir>` as an environment variable named `CMP_ECOM_DIR_CONFIG`. Only the command `ecom` is necessary to start ECOM in this instance. ECOM looks on the command line first, and if no path is specified, it looks for this environment variable to provide this information. If this value is not set in either location, ECOM will terminate.

Running ECOM as a daemon

To run ECOM as a daemon, use the `-d` command line option. For example:

```
./ecom -d -c <ecom_conf_dir>
```

Running ECOM with non-root privileges

To use a port number less than 1024 on a Linux system, ECOM must start with root privileges. However, after port binding, it does not need to continue as a root user. You can start ECOM as a root user but then reduce its privileges with the following command:

```
./ecom -u <username>
```

The -u command-line flag instructs ECOM to reduce its privileges to those of the specified username after port binding. The specified username must be a valid operating system user and must have read, write, and execute permissions for <ECOM_install_dir>.

Note: If you run ECOM with a non-root user account, it is recommended that you set the non-root user as the owner of <ECOM_install_dir> or have the directory's permissions set to a group that has the user as the only member. These security settings help to prevent tampering with ECOM files.

Additionally, the specified username must have read and write permissions for the following ECOM files and directories:

```
<ECOM_install_dir>/bin/*
<ECOM_install_dir>/conf/cst/*
<ECOM_install_dir>/conf/cst/LocalDirectoryData.xml
<ECOM_install_dir>/conf/cst/RoleData.xml
<ECOM_install_dir>/log/*
<ECOM_install_dir>/conf/ssl/*
```

Note: The -u command line parameter does NOT work for non-Linux\UNIX hosts.

Windows

To start ECOM on a Windows-based platform:

1. Open a command line window (`cmd.exe`) to obtain a DOS prompt.
2. Navigate to `<ECOM_install_dir>\bin`.
3. Execute the command:
   ```
   ECOM.exe -c <ecom_conf_dir>
   ```

   where `<ecom_conf_dir>` is the full path to the directory containing the file `ECOM_settings.xml`. ECOM will start in the command window.

Note: You can also set the path for `<ecom_conf_dir>` as an environment variable named `CMP_ECOM_CONFIG_DIR`. Only the command `ecom.exe` is necessary to start ECOM in this instance. ECOM looks on the command line first, and if no path is specified, it looks for this environment variable to provide this information. If this value is not set in either location, ECOM will terminate.

Starting ECOM as a Windows service

ECOM can also be run as a Windows service by following these steps:

1. Open a command line window (`cmd.exe`) to obtain a DOS prompt.
2. Navigate to `<ECOM_install_dir>\bin`.
3. Execute the command:

   ```
   sm_service install --force --name=ECOM.exe
     --description=ECOM  --startmode=runonce
     <ECOM_install_dir>\bin\ECOM.exe
   ```
4. Execute the command:

   ```
   sm_service start ECOM.exe
   ```

   to start the ECOM service.

Note: If ECOM fails to start as a Windows service, check the Windows Event Viewer for related errors.

Stopping ECOM as a Windows service

To stop a running ECOM service:

1. Open a command line window (`cmd.exe`) to obtain a DOS prompt.
2. Navigate to `<ECOM_install_dir>\bin`.
3. Execute the command:

   ```
   sm_service stop ECOM.exe
   ```
4. To also uninstall ECOM as a service, execute the command:

   sm_service remove ECOM.exe

LARGEADDRESSSAWARE linker support

On 32-bit Windows builds, ECOM is now linked with the /LARGEADDRESSAWARE linker flag. When running a 32-bit ECOM on a 32-bit Windows host:

- If the Windows host running ECOM does not have the /3GB flag in its boot.ini file, then ECOM's memory management behavior does not change. ECOM can use a maximum of 2GB of virtual memory. The kernel uses the other 2GB.
- If the 32-bit Windows host has the /3GB flag set in boot.ini, the kernel is limited to 1GB of virtual memory, and ECOM will be able to use up to 3GB of virtual memory.
- When running a 32-bit ECOM on a 64-bit Windows host, ECOM will be able to use up to 4GB of virtual memory instead of only 2GB.

  On some Windows 32-bit hosts, depending on the system hardware, enabling the /3GB flag in boot.ini may not work and Windows may not even boot afterward.

Accessing the ECOM Administration Web Server

ECOM contains a web server used for administering logs and security. It can also be used to deliver HTML content and other files, such as JAR files for Java client user interfaces. By default, the server can be accessed from:

```
https://<ManagementIPAddress>:<port>/ecomconfig
```

where `<port>` is a secure port as defined in `Port_settings.xml`.

Note: The parameter `ECOMConfigPageEnabled` in ECOM_settings.xml must be set to true (the default) to enable access to the ECOM Administration Web Server.

The web server requires authentication. If you are accessing ECOM for the first time, you can login with the default credentials of:

Username: LocalDir/admin

Password: #1Password

Note: EMC recommends changing this user ID/password combination after initially starting ECOM.

The web interface contains additional instructions for its usage. The following features are currently provided after authentication:

- Display logs
- Add and delete local user accounts
- Change local user account passwords

- Define an LDAP directory server for authentication and authorization
- Map operating system users and groups to ECOM's defined roles (Windows only)
- Generate a self-signed SSL certificate
- Create an SSL Certificate Signing Request (CSR) to send to an authority
- Import a signed SSL certificate for use by ECOM

Delivering web content

Web content can be delivered through ECOM via HTTP by placing it in the www folder of ECOM.

GZIP and Pack200 compression support for JAR files

Some ECOM clients use its web server to deliver JAR files for applet-based UIs. ECOM supports the GZIP and Pack200 compression formats to support faster delivery of JAR files to consuming clients. To request GZIP or Pack200 versions of JAR files from ECOM, clients must set the Accept-Encoding (AE) field in the HTTP header to "pack200-gzip" or "gzip", indicating to ECOM that the client application can handle the pack200-gzip or gzip format, respectively. ECOM searches for the requested JAR file with .pack.gz or .gz file extension and returns the located file. ECOM also sets the response header Content-Encoding (CE) field to "pack200-gzip" , "gzip", or NULL depending on the type of file that is being sent, and optionally may set the Content-Type (CT) to "application/Java-archive". Therefore, by inspecting the CE field, the requesting application can apply the corresponding transformation to restore the original JAR file.

Stopping ECOM

There are several ways to stop the ECOM process depending on the platform it is running on. See your platform's operating system instructions for specific details about stopping processes, services, and daemons.

IMPORTANT

There is currently no recommended, graceful way to shut down ECOM.

Stopping ECOM from the command line

The most common method for stopping the ECOM process is from the command line. If you have access to the command line shell that started the ECOM process, you can stop it with a **Control-C** keyboard input.

Stopping ECOM on Linux

If you plan to shutdown ECOM gracefully with a SIGTERM on Linux, you can specify the amount of time it will wait for shutdown completion before the process is killed. This setting is defined in ECOM_settings.xml as:

```
<ECOMSetting Name="ShutdownTimeout" Type="uint32"
Value="20"/>
```

ShutdownTimeout is specified in seconds. If the process is killed, ECOM will log this message in cimomlog.txt:

```
Cimom Startup-PostStartup section has timed out, shutting
down.
```

Checking the installed version of ECOM

ECOM version information, including build date and time, can be displayed from the command line using the '-v' parameter. For example:

```
<ECOM_install_dir>\bin\ECOM -v
```

yields the response

```
ECOM Version 9.9.0.0 (Build Date & Time: Feb 11 2009,
17:59:57)
```

This versioning information can also be accessed externally through CIM-XML from the CIM_SoftwareIdentity.versionString property or from the ECOM Administration Web Server.

Running ECOM under a debugger

ECOM can be run under a debugger. After installing and configuring a debug version of ECOM available from the latest ECOM Toolkit distribution, follow these instructions.

Debugging ECOM initialization

If you want to debug the ECOM process, you must use the debug version of ECOM included in the ECOM Toolkit. If you have OSLS plug-ins attached, you should also use their debug versions as well.

Execute ECOM debug using the following commands.

1. Open a shell to access a command line.
2. Navigate to `<ECOM_install_dir>/bin`.

3. Execute the command:

   ```
   ECOM.exe -p -c <ecom_conf_dir>
   ```

   ECOM returns a message similar to:

   ```
   PID 1228 waiting for attach before program load; set
   predebug = 0 to continue
   ```

4. You can now attach to the ECOM process in a debugger, such as Microsoft Visual Studio, and set your breakpoints. To allow ECOM to continue initialization, the variable `predebug` must be changed to a value of 0.

Debugging ECOM normal operations

If you want to attach to the ECOM process after startup, you must use the debug version of ECOM included in the ECOM Toolkit. If you have OSLS plug-ins attached, you should also use their debug versions as well.

Execute ECOM debug using the following commands.

1. Set the following environment variables to include important ECOM debug libraries.

   - PATH (Windows only) — This variable must include the following ECOM folders:
     - lib
     - thirdparty
     - adapters
   - SM_NOLICENSE (Windows only) — Set to a value of 1.
   - SM_LIBPATH (Windows only) — This variable must include the following ECOM folders:
     - thirdparty
     - adapters
     - providers
   - SM_HOME (Windows only) — This variable must include the path represented by `<ECOM_install_dir>`.
   - CMP_ECOM_DIR_BASE (Windows only) — This variable must match the value of SM_HOME.
   - NT_SYMBOL_PATH (Windows only)—This variable must include the following ECOM debug folders:
     - symbols/exe
     - symbols/dll
     - providers

  - bin
- SM_DEBUG (Linux only) — Set to a value of 1.
- SM_DEBUGGER (Linux only) — The variable identifies the path to **gdb**.

IMPORTANT

You must set these variables manually because the environment variable script used by ECOM, `<ECOM_install_dir>/bin/runcmd`, is not run as part of the debug process.

2. Open a shell to access a command line.
3. Navigate to `<ECOM_install_dir>\bin\system`.
4. Execute the command:
   ```
   ECOM.exe -c <ecom_conf_dir>
   ```

You can now attach to the ECOM process in a debugger, such as Microsoft Visual Studio.

Reviewing ECOM log files

ECOM and its core providers log information to the following files located in `<ECOM_install_dir>\log by default`:

- `cimomlog.txt`—Contains ECOM operational messages and security messages from the enabled security plug-in. To assist in the debugging and review of ECOM operations, the thread ID for each entry logged in cimomlog.txt is available. Examples of thread IDs in messages are below in red.

```
11-Jan-2010 12:16:16.584 -2464-T-
NAVHTTPServerListener: Received connection from
lglom183.lss.emc.com[172.23.120.183]:54653 (on
USENSANEPD1C.corp.emc.com[168.159.66.73]:5988)
```

  On a Linux platform, the entry looks like:

```
13-Jan-2010 08:52:14.551 -3058752416-T-
NAVHTTPServerListener: Received connection from
usensanepd1c.corp.emc.com[0:0:0:0:0:FFFF:A89F:4249]:1
358 (on
lglom183.lss.emc.com[0:0:0:0:0:FFFF:AC17:78B7]:59088)
```

- `LegacyIndicationsLog.txt`—This file is not currently used by ECOM.
- `securitylog.txt`—Contains security-related messages from the enabled security plug-in.
- HTTP_trace.log—Contains HTTP input messages (requests), output messages (responses), indication registrations, and indications sent by ECOM.

You can change how ECOM logs information to `cimomlog.txt`, `securitylog.txt`, and `HTTP_trace.log` in the configuration file `Log_settings.xml`.

IMPORTANT

You must restart ECOM for updated log settings to take effect.

The following parameters are defined within the file:

- `LogDir` — The directory to write log files relative to the installation directory of ECOM. The default directory is `log`.
- `Mode` — This value defines whether ECOM writes log entries (`Enabled`) or does not write them (`Disabled`).

- Severity — The minimum severity level for ECOM to log to cimomlog.txt. ECOM writes log messages of this level and higher. Possible values in descending order of severity are:
  - NAVI_CRITICAL
  - NAVI_ERROR
  - NAVI_WARNING
  - NAVI_INFORMATION
  - NAVI_TRACE (default)
- MaxSize — Maximum size of cimomlog.txt in bytes. When this value is exceeded, ECOM begins writing a new log file. The previous log file remains on the file system. ECOM only preserves the last three logs. Older log files are deleted.
- Facility — This legacy parameter is not used and should not be modified.
- AbortOnLogInitErrors — Whether ECOM should abort startup if log file initialization fails. Setting this boolean value to true tells ECOM to abort startup if it cannot initialize its log files (cimomlog.txt and security_log.txt). This generally occurs when the ECOM process does not have permission to write to the directory specified by LogDir. A value of false allows ECOM to ignore these errors and continue startup. The default value is true.

HTTP trace log settings

Log_settings.xml also defines how ECOM logs HTTP trace statements (Figure 17 on page 69) that are useful for debugging problem resolution. Using HTTP tracing, you can examine CIM-XML requests and responses in addition to HTTP requests made to the ECOM Administration Web Server or to custom web providers.

- HTTPTraceDest — The destination file for HTTP trace messages. Possible values are TRACE, which sends all messages into a separate file named HTTP_trace.log in the log directory, or LOG, which puts all trace messages in cimomlog.txt. The default is TRACE.
- HTTPTraceOutput — Set this value to true to start tracing HTTP responses sent by ECOM. The default value is false.
- HTTPTraceOutputBufSize — The size in bytes of the buffer used to hold the HTTP response trace data before it is written to the destination file. If the buffer fills up before the output message is complete, it will be written out and assigned a sequential segment number starting at 1. A value of zero, which is the default, sets a

value of 1MB or ¼ of the trace file maximum size (set by `HTTPTraceMaxSize`), whichever is smaller. Positive, non-zero values must be between 1024 bytes and 1 MB.

- `HTTPTraceInput` — Set this value to `true` to start tracing HTTP request sent to ECOM. The default value is `false`.
- `HTTPTraceInputBufSize` — The size in bytes of the buffer used to hold the HTTP request trace data before it is written to the destination file. If the buffer fills up before the input message is complete, it will be written out and assigned a sequential segment number starting at 1. A value of zero, which is the default, sets a value of 1MB or ¼ of the trace file maximum size (set by `HTTPTraceMaxSize`), whichever is smaller. Positive, non-zero values must be between 1024 bytes and 1 MB.
- `HTTPTraceMaxSize` — The maximum size of the trace file in bytes. This setting does not limit the size of `cimomlog.txt` if `HTTPTraceOutput` is set to `LOG`.
- `TraceIndications` — Set this value to `true` to start tracing HTTP indications (output) and indication subscriptions (input). The default value is `false`.
- `TraceIndicationsBufSize` — The size in bytes of the buffer used to hold the HTTP indication trace data before it is written to the destination file. If the buffer fills up before the message is complete, it will be written out and assigned a sequential segment

number starting at 1. A value of zero, which is the default, sets a value of 1MB or ¼ of the trace file maximum size (set by `HTTPTraceMaxSize`), whichever is smaller. Positive, non-zero values must be between 1024 bytes and 1 MB.

```
22-Oct-2007 14:31:02.984 -T- NAVSocket: HTTP output:
------------------------------ Output trace to 127.0.0.1:2559 Segment 1
  ------------------------------
HTTP/1.1 200 OK
Content-Type: application/xml; charset="utf-8"
CIMOperation: MethodResponse
Transfer-encoding: chunked

0F7B
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
<MESSAGE ID="1001" PROTOCOLVERSION="1.0">
<SIMPLERSP>
<IMETHODRESPONSE NAME="EnumerateQualifiers">
<IRETURNVALUE>
  <QUALIFIER.DECLARATION NAME="Abstract" TYPE="boolean" OVERRIDABLE="true"
    TOSUBCLASS="false" TRANSLATABLE="false" >
    <SCOPE CLASS="true" ASSOCIATION="true" INDICATION="true" PROPERTY="false"
      REFERENCE="false" METHOD="false" PARAMETER="false" ></SCOPE>
    <VALUE>false</VALUE>
  </QUALIFIER.DECLARATION>
  <QUALIFIER.DECLARATION NAME="Aggregate" TYPE="boolean" OVERRIDABLE="false"
    TOSUBCLASS="true" TRANSLATABLE="false" >
    <SCOPE CLASS="false" ASSOCIATION="false" INDICATION="false"
      PROPERTY="false" REFERENCE="true" METHOD="false" PARAMETER="false"
      ></SCOPE>
    <VALUE>false</VALUE>
  </QUALIFIER.DECLARATION>
  <QUALIFIER.DECLARATION NAME="Aggregation" TYPE="boolean"
    OVERRIDABLE="false" TOSUBCLASS="true" TRANSLATABLE="false" >
    <SCOPE CLASS="false" ASSOCIATION="true" INDICATION="false"
      PROPERTY="false" REFERENCE="false" METHOD="false" PARAMETER="false"
      ></SCOPE>
    <VALUE>false</VALUE>
  </QUALIFIER.DECLARATION>
  <QUALIFIER.DECLARATION NAME="ArrayType" TYPE="string" OVERRIDABLE="false"
    TOSUBCLASS="true" TRANSLATABLE="false" >
    <SCOPE CLASS="false" ASSOCIATION="false" INDICATION="false"
      PROPERTY="true" REFERENCE="false" METHOD="false" PARAMETER="true"
      ></SCOPE>
  ...
```

Figure 17 Sample HTTP trace log data

Testing ECOM deployments

Once you have started ECOM from the command-line, you may want to test its response within your network. This section presents procedures and tools to help you quickly test ECOM.

Checking ECOM response

The following sections describe basic tools and tests you can run to ensure proper operation of an installed ECOM instance.

SLP response

See Appendix A, "SLP Configuration" for information about how to configure the ECOM SLP provider and test its response.

CIM-XML over HTTP response

Checking for a response from the ECOM Server Profile Provider, which is configured to run by default in ECOM, is a good way to check for proper ECOM CIM-XML response. See "Testing response" on page 124 in Appendix B for instructions on testing the ECOM Server Profile Provider.

Checking SMI-S compliance

Check the generic response of ECOM to an SLP multicast or a directed CIM-XML message using the SMI-S Test Suite available through the SMI-S Compliance Testing Program (CTP).

Obtaining further information about ECOM

Assistance is available for EMC developers using ECOM as follows:

- Support: Common Management Group Forum at http://cmgportal.lss.emc.com/
- Request for developer consultation: Send an email to CMGSupport@emc.com

3 ECOM Application Security

ECOM supports authentication, authorization, and accounting (AAA) to protect managed resources. The following sections discuss how to set up and configure ECOM application security:

AAA in ECOM

ECOM supports authentication, authorization, and accounting to secure exposed data.

Authentication

ECOM receives client credentials using one of the following HTTP(S) Authentication protocols:

- HTTP BASIC Authentication, as described in RFC 2617.
- HTTP Digext Authentication as described in RFC 2617.
- SPNEGO-based Kerberos and NTLM HTTP Authentication in Microsoft Windows. This is described in RFC 4559.

With Basic authentication, usernames and passwords must be Base64 encoded. With other forms of authentication, credentials are passed through to the ECOM security plugin as received from the client.

Once user credentials have been received, ECOM processes them in a pluggable security architecture that routes authentication requests to the configured security plug-in.

The authentication method used by ECOM is configurable. The ECOM configuration file **security_settings.xml** contains the parameter `HTTPChallengeMechanism`. Set this parameter to one or more of the following values:

- Basic
- Digest
- Negotiate

You can specify more than one authentication method, as follows:

```
<ECOMSetting Name="HTTPChallengeMechanism"
Type="string"  Value="Basic, Digest, Negotiate"/>
```

Negotiate support, as implemented by ECOM, is described in Figure 18.

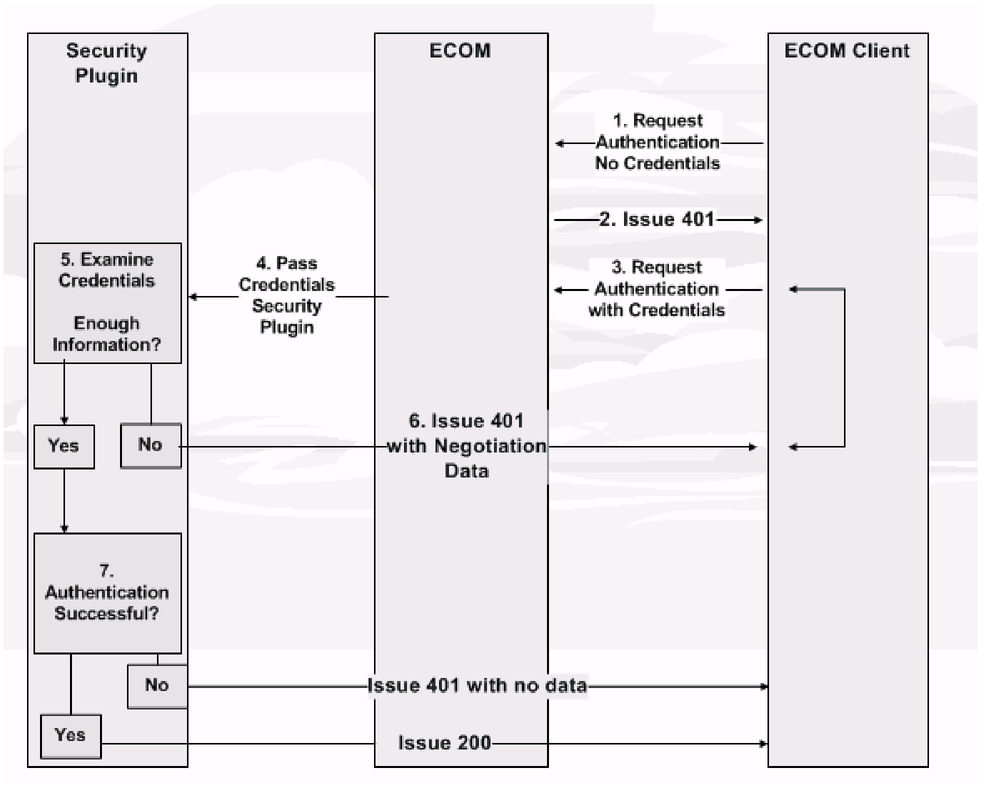
Security Plugin
ECOM
ECOM Client
1. Request Authentication No Credentials
2. Issue 401
3. Request Authentication with Credentials
4. Pass Credentials Security Plugin
5. Examine Credentials Enough Information?
Yes
No
6. Issue 401 with Negotiation Data
7. Authentication Successful?
No
Yes
Issue 401 with no data
Issue 200

Figure 18 **Negotiate Authentication**

Cookie Support

ECOM Supports Cookies in compliance with http://tools.ietf.org/html/rfc6265.

Cookie support is independent of the type of authentication you choose.

Cookies expire 60 seconds after the last request. If more than 60 seconds has passed since the last request, clients will have to re-authenticate.

If the client browser suppresses cookies, the client will have to authenticate each request.

Authorization

ECOM implements Role-Based Access Control (RBAC) to authorize authenticated client requests. Security in ECOM is defined on a per provider basis, allowing access to be limited at the class, instance, method, and property level of each provider. This allows the information and operations available through a provider or OSLS plug-in attached to ECOM to be limited on an operation by operation basis according to the roles assigned to an authenticated user.

Roles are directly assigned to CIM classes, methods, and properties in the registration file (*_prg.xml) for each provider attached to ECOM. Figure 19 on page 76 provides an example qualifier that restricts access to a class to a defined set of roles.

Note: Access for concrete classes must be specifically defined in a provider registration file. Access is not allowed through inherited parent classes. For example, an authenticated client does not automatically have access to CIM_Capabilities, as shown in Figure 19 on page 76, if it meets the role requirements for its parent class CIM_ManagedElement.

```
...
<Class Name="CIM_Capabilities" BaseClass="CIM_ManagedElement">
<Qualifier name="Dynamic" />
      <Qualifier name="Provider">
        <Value>ServerProfileProvider</Value>
      </Qualifier>
      <Qualifier name="Roles">
        <Value>administrator,manager,monitor,provider</Value>
      </Qualifier>
...
```

Figure 19 Roles required to access class CIM_Capabilities in ServerProfileProvider_prg.xml

Note: If a client attempts to access an object that is not managed by ECOM when authentication is enabled, ECOM will respond with an authorization failure. Therefore, when you receive this response, ensure that the authentication string is correct. If the string is accurate, check to make sure that the class or instance in the request is managed by ECOM.

ECOM roles

The roles shown in Table 11 on page 77 allow authenticated users to access specific sets of functionality within ECOM. These roles are predefined in all ECOM distributions and should not be changed.

Table 11 **ECOM roles**

Role name	Definition
administrator	An administrator has access to all data and functionality available through ECOM and its attached providers.
manager	A manager can access the data and functionality exposed by ECOM but cannot change ECOM configuration itself.
monitor	A monitor can view all system data exposed by ECOM but cannot perform any operations against managed resources or against ECOM.
securityadministrator	A security administrator is able to perform security tasks but is not able to access data or functionality exposed by ECOM.

ECOM loads these roles from the file <ECOM_install_dir>/providers/ECOMRoles.xml.

You can add your own custom roles to ECOMRoles.xml. You can then use them with your ECOM provides and OSLS plug-ins. Figure 20 on page 78 demonstrates ECOMRoles.xml with a custom role for an application administrator added.

Note: ECOM must be (re)started to load and use new roles.

```
<?xml version="1.0" ?>
<DefinedRoles>
  <Role Name="administrator" Description="User will have access to all
administrative and management interfaces and data.">
    <Enum Value="0"/>
  </Role>
  <Role Name="manager" Description="User will be able to see all storage
system data and perform all storage management operations.">
    <Enum Value="1"/>
  </Role>
  <Role Name="monitor" Description="User will be able to see all storage
system data, but not perform any operations.">
    <Enum Value="2"/>
  </Role>
  <Role Name="securityadministrator" Description="User will only be able to
perform security tasks. They will not be able to see storage data.">
    <Enum Value="4"/>
  </Role>
<Role Name="application_admin" Description="This user will be able to view
and perform operations against host-based applications.">
    <Enum Value="5"/>
  </Role>
</DefinedRoles>
```

Figure 20 ECOMRoles.xml with a custom role (application_admin) added

Mapping LDAP groups to ECOM roles

To authorize users from an LDAP directory, ECOM must be able to map its roles to the LDAP groups to which a user belongs. Appendix C, "CST Security Plug-in" provides additional information required to map groups from an LDAP authority defined through the CST to ECOM roles.

Mapping operating system users and groups to ECOM roles (Windows only)

"Role mapping for Windows operating system users and groups (OSLogin authority)" on page 133 provides instructions for mapping Windows local operating system and domain users and groups to ECOM roles using the ECOM SDK.

Mapping roles to OSLS plug-ins

"Applying ECOM security to OSLS providers" on page 130 provides more information about how to map the resources and functionality exposed by OSLS plug-ins to ECOM's security roles through MOF qualifiers.

Accounting

ECOM logs security-related information to the file securitylog.txt, as shown in Figure 21 on page 79, if the configuration parameter

SecurityLoggingEnabled is set to true. In the Figure 21 example, user access to class CIM_System is challenged, and the required authentication succeeds. Whether security logging is enabled or not, security-related information is always logged to the file cimomlog.txt. The easiest way to view the contents of log files is through the **ECOM Administration Web Server**. "Accessing the ECOM Administration Web Server" on page 61 provides more information.

```
09-JUN-2007 16:21:38.163 -T- ECOM Webserver: Scanning the http(s) request
      obtained
09-JUN-2007 16:21:38.163 -I- ECOM: Authenticating incoming request
09-JUN-2007 16:21:38.163 -I- ECOM: No credentials received. Challenging
      client for credentials
09-JUN-2007 16:21:38.163 -T- ECOM Webserver: Scanning the http(s) request
      obtained
09-JUN-2007 16:21:38.163 -I- ECOM: Authenticating incoming request
09-JUN-2007 16:21:38.163 -T- Security: Starting new transaction
09-JUN-2007 16:21:38.163 -T- Security: Ending Transaction
09-JUN-2007 16:21:38.163 -I- ECOM: Authenticated client "admin"
09-JUN-2007 16:21:38.163 -T- Security: Starting new transaction
09-JUN-2007 16:21:38.163 -I- ECOM: Checking authorization for admin on
      CIM_System with Security provider.
09-JUN-2007 16:21:38.163 -I- ECOM SecurityProvider: Checking access for user
      admin to CIM_System
09-JUN-2007 16:21:39.945 -I- ECOM SecurityProvider: class CIM_System exists
      in the table and accessible to administrator. Access granted.
09-JUN-2007 16:21:39.945 -I- ECOM SecurityProvider: User admin has been
      granted access to CIM_System
09-JUN-2007 16:21:39.945 -T- Security: Ending Transaction
09-JUN-2007 16:21:39.945 -I- ECOM: Authorization succeeded for admin on
      CIM_System
```

Figure 21 Sample security log entries in securitylog.txt

General settings

General ECOM security options can be configured in the file `Security_settings.xml`, as shown in Figure 22 on page 80, which can be found in <ECOM_install_dir>\conf.

```
<ECOMSettings>

  <ECOMSetting Name="CIMRequest_AuthenticationEnabled" Type="boolean"
    Value="false"/>
  <ECOMSetting Name="NonCIMRequest_AuthenticationEnabled" Type="boolean"
    Value="false"/>
  <ECOMSetting Name="PluginName" Type="string"  Value="CSTSecurityPlugin"/>
  <ECOMSetting Name="CSTConfigDirectory" Type="string" Value="conf/cst"/>
  <ECOMSetting Name="LocalUserAccountsDirectory" Type="string"
    Value="providers"/>
  <ECOMSetting Name="EncryptedRoleFiles" Type="boolean" Value="false"/>
  <ECOMSetting Name="EncryptionAlgorithm" Type="string" Value="RC4"/>
  <ECOMSetting Name="HTTPRealmIdentifier" Type="string" Value="ECOM_SERVER"/>

  <! -- Security logging settings -->
  <ECOMSetting Name="SecurityLoggingEnabled" Type="boolean" Value="true"/>
  <ECOMSetting Name="SecurityLogDirectory" Type="string" Value="log"/>
  <ECOMSetting Name="CSTLoggingEnabled"Type="boolean"  Value="true"/>

  <!-- SSL settings -->
  <ECOMSetting Name="SSLClientAuthentication" Type="string"  Value="None"/>
  <ECOMSetting Name="CertificatesDirectory" Type="string" Value="conf/ssl"/>
  <ECOMSetting Name="SSLConfigFile"Type="string" Value="cert_config.cnf"/>
  <ECOMSetting Name="SSLCertFile"Type="string" Value="ecomtls.crt"/>
  <ECOMSetting Name="SSLKeyFile"Type="string" Value="ecomtls.pk"/>
  <ECOMSetting Name="SSLCAFile"Type="string" Value="ecomtls.ca"/>
  <ECOMSetting Name="SSLUserGeneratedCsrFile "Type="string"
    Value="ecomtls_user.csr"/>
  <ECOMSetting Name="SSLAutoGeneratedCsrFile "Type="string"
    Value="ecomtls.csr"/>
  <ECOMSetting Name="SendSSLClientCertificate" Type="boolean" Value="true" />

  <ECOMSetting Name="ManageCSTLockBox"  Type="boolean"  Value="false"/>
  <ECOMSetting Name="ExternalConnectionLimit" Type="uint32" Value="100" />
  <ECOMSetting Name="CSTLoggingEnabled" Type="boolean" Value="true" />
  <ECOMSetting Name="FIPSMode" Type="boolean" Value="false" />
  <ECOMSetting Name="GenerateCertificate" Type="boolean" Value="true" />
  <ECOMSetting Name="WSTrustSessionTimeout" Type="uint32" Value="300" />

</ECOMSettings>
```

Figure 22 Sample Security_settings.xml contents

The general security-related parameters in Security_settings.xml are:

- CIMRequest_AuthenticationEnabled — A value of true requires authentication and authorization for incoming CIM-XML requests. A value of false allows ECOM to respond to all CIM-XML requests without authentication. The default value is false.
- NonCIMRequest_AuthenticationEnabled— A value of true requires authentication and authorization for all non-CIM-XML incoming requests, such as HTTP requests. A value of false allows ECOM to respond to all requests without authentication. The default value is false.

Note: The ECOM Administration Web Server will always require authentication irrespective of the value of `NonCIMRequest_AuthenticationEnabled`.

- EncryptedRoleFiles — A value of true tells ECOM to look for encrypted roles files for attached providers in the format *_RolesEncrypted.xml in the conf directory. The default value of false instructs ECOM to use unencrypted roles files of the format *_Roles.xml in the conf directory. ECOM does not automatically encrypt roles files for providers. You must encrypt these manually with ECOM's Encrypt tool, which is included in the bin directory. "Using the Encrypt command line tool" on page 94 provides details for using this tool. Some of the default roles files you may need to encrypt, depending on your ECOM configuration, are:
  - SecurityPlugin_RolesEncrypted.xml
  - DirectoryProvider_RolesEncrypted.xml
  - ECOMRolesEncrypted.xml
  - PersistenceProvider_RolesEncrypted.xml
  - SecurityProvider_RolesEncrypted.xml
  - IndicationSubscriptionProvider_RolesEncrypted.xml
  - IndicationSubscriptionProvider/IndicationSubscriptionProvider_RolesEncrypted.xml
  - RepositoryProvider_RolesEncrypted.xml
  - SampleProvider_RolesEncrypted.xml
  - ServerProfileProvider_RolesEncrypted.xml
  - SLPProvider_RolesEncrypted.xml

- EncryptionAlgorithm — ECOM supports two file encryption algorithms for security. They can be specified by values of AES or RC4. RC4 is the default algorithm.
- FIPSMode — Defines whether ECOM should be started in FIPS 140-compliant mode. ECOM uses AES by default in FIPS-compliant mode and RC4 in non-FIPS-compliant mode.
- HTTPRealmIdentifier — A unique string that identifies this ECOM instance. This string is sent during the HTTP authentication challenge process. Consult the SMI-S 1.2 specification for a recommended format for this string.
- SecurityLogDirectory — This value defines the directory location relative to <ECOM_install_dir> in which ECOM will write its security log securitylog.txt. You must create this directory before ECOM startup if you change it from the ECOM default.
- SecurityLoggingEnabled — The default value of true instructs ECOM to log security-related events to securitylog.txt in addition to cimomlog.txt. A value of false disables security logging to securitylog.txt but continues its logging to cimomlog.txt. "Reviewing ECOM log files" on page 66 provides more information about ECOM logs. Figure 21 on page 79 provides a security log example.
- CSTLoggingEnabled — This value defines whether the CST Security Plug-in will write CST logs, which may impact performance. The default value is true, enabling CST logging.
- ManageCSTLockBox — The default value of false instructs ECOM to use a local, pre-installed and pre-initialized CST lockbox. A value of true instructs ECOM to initialize a new lockbox for its own use.
- GenerateCertificate — The default value of false instructs ECOM to use the existing SSL security certificate for secure communication. A value of true instructs ECOM to generate a new security certificate at startup.
- ExternalConnectionLimit - Before ECOM 2.6.6, ECOM imposed a limit of 100 active client connections. In ECOM 2.6.6, this parameter was added to security_settings.xml to allow administrators to explicitly set the maximum number of connections allowed by ECOM.
- ExternalConnectionLimitPerHost - This parameter limits the number of connections that can be established from a single host. This parameter is used in conjunction with

ExternalConnectionLimit. For instance, if ExternalConnectionLimit=20 and ExternalConnectionLimitPerHost=5, than ECOM could have 4 hosts with 5 connections each, or 20 with one connection each. However, no host would be allowed 6 or more connections.

- WSTrustSessionTimeout - This property sets the amount of time (in seconds) for ECOM to maintain the valid a WS-Trust session for the interactive authentication process. This timeout established the amount of time the client is allocated to answer the authentication challenge questions. If the client does not respond and the timeout expires, subsequent attempts to provide credentials using the same session will result in ECOM returning an error encoded in a SOAPFault envelope, as shown in Figure 23 on page 83.

```
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
xmlns:wst14="http://docs.oasis-open.org/ws-sx/ws-trust/200802">

   <s:Header/>
      <s:Body>
         <s:Fault>
             <s:Code>
                <s:Value>s:Sender</s:Value>
                   <s:Subcode>
                <s:Value>wst14:InvalidRequest</s:Value>
                   <s:Subcode>
             </s:Code>
         <s:Reason>
             <s:Text xml:lang="en-US">The request was invalid or malformed
             </s:Text>
         </s:Reason>
              <s:Detail>No ContextData@RefId element at
              InteractiveChallengeResponse or could not find the
              ContextData@RefId supplied.</s:Detail>
         </s:Fault>
      </s:Body>
</s:Envelope>
```

Figure 23 SOAPFaultEnvelope - Interactive Challenge Response Timeout

Note: If Security_settings.xml cannot be found or is not accessible, ECOM will not be able to determine the security plug-in to load and will not start up.

- SSLCipherSuites - This option controls the supported cipher suites of the SSL connection. Cipher suites are processed in order so the most preferred cipher suites should appear at the beginning of the list. For example,

  ```
  <ECOMSetting Name="SSLCipherSuite" Type="string"
  Value="EDH-DSS-AES256-SHA:AES256-SHA:DES-CBC3-SHA:RC4
  -MD5" />
  ```

  where EDH-DSS-AES256-SHA is the preferred cipher suite and RC4-MD5 is the least preferred. Use colons ":" to separate the options. When operating in FIPS 140-2 mode only FIPS 140-2 compliant algorithms are available. Unrecognized cipher suites are ignored and no error occurs indicating this.

  The cipher suites supported by ECOM are as follows:

  - NSA Suite B Compliant Cipher Suites (defined in RFC 5289)
  - NSA Suite B Transitional Cipher Suites (defined in RFC 4492)
  - Non-Suite B, AES GCM Cipher Suites (defined in RFC 5288)

Additionally, Table 12 on page 84 describes the Non-Suite B Cipher Suites supported by ECOM.

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Cipher Suite Name	AECDH-NULL-SHA
Standard Cipher Suite Name	TLS_ECDH_anon_WITH_NULL_SHA
Key Exchange Algorithm	Anonymous Elliptic Curve Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	None
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	AECDH-RC4-SHA
Standard Cipher Suite Name	TLS_ECDH_anon_WITH_RC4_128_SHA
Key Exchange Algorithm	Anonymous Elliptic Curve Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	RC4 with 128-bit key
Message Digest Algorithm	SHA-1

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

FIPS	YES
Cipher Suite Name	AECDH-DES-CBC3-SHA
Standard Cipher Suite Name	TLS_ECDH_anon_WITH_3DES_EDE_CBC_SHA
Key Exchange Algorithm	Anonymous Elliptic Curve Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	Triple DES (EDE) with 168-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	AECDH-AES128-SHA
Standard Cipher Suite Name	TLS_ECDH_anon_WITH_AES_128_CBC_SHA
Key Exchange Algorithm	Anonymous Elliptic Curve Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	AECDH-AES256-SHA
Standard Cipher Suite Name	TLS_ECDH_anon_WITH_AES_256_CBC_SHA
Key Exchange Algorithm	Anonymous Elliptic Curve Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-DSS-AES256-SHA
Standard Cipher Suite Name	TLS_DHE_DSS_WITH_AES_256_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-DSS-AES256-SHA256
Standard Cipher Suite Name	TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-256
FIPS	YES
Cipher Suite Name	EDH-RSA-AES256-SHA
Standard Cipher Suite Name	TLS_DHE_RSA_WITH_AES_256_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	RSA
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-RSA-AES256-SHA256
Standard Cipher Suite Name	TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	RSA
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-256
FIPS	YES
Cipher Suite Name	AES256-SHA
Standard Cipher Suite Name	TLS_RSA_WITH_AES_256_CBC_SHA

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	AES256-SHA256
Standard Cipher Suite Name	TLS_RSA_WITH_AES_256_CBC_SHA256
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	AES with 256-bit key in CBC mode
Message Digest Algorithm	SHA-256
FIPS	YES
Cipher Suite Name	EDH-DSS-AES128-SHA
Standard Cipher Suite Name	TLS_DHE_DSS_WITH_AES_128_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-DSS-AES128-SHA256
Standard Cipher Suite Name	TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-256
FIPS	YES

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Cipher Suite Name	EDH-RSA-AES128-SHA
Standard Cipher Suite Name	TLS_DHE_RSA_WITH_AES_128_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	RSA
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-RSA-AES128-SHA256
Standard Cipher Suite Name	TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	RSA
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-256
FIPS	YES
Cipher Suite Name	AES128-SHA
Standard Cipher Suite Name	TLS_RSA_WITH_AES_128_CBC_SHA
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	AES128-SHA256
Standard Cipher Suite Name	TLS_RSA_WITH_AES_128_CBC_SHA256
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	AES with 128-bit key in CBC mode

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Message Digest Algorithm	SHA-256
FIPS	YES
Cipher Suite Name	EDH-DSS-DES-CBC3-SHA
Standard Cipher Suite Name	SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA
Encryption Algorithm	Triple DES (EDE) with 168-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-RSA-DES-CBC3-SHA
Standard Cipher Suite Name	SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA TLS_DHE_RSA_WITH_3DES_EDE_CBC_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	RSA
Encryption Algorithm	Triple DES (EDE) with 168-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	DES-CBC3-SHA
Standard Cipher Suite Name	SSL_RSA_WITH_3DES_EDE_CBC_SHA TLS_RSA_WITH_3DES_EDE_CBC_SHA
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	Triple DES (EDE) with 168-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	EDH-DSS-RC4-SHA

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Standard Cipher Suite Name	TLS_DHE_DSS_WITH_RC4_128_SHA
Key Exchange Algorithm	Ephemeral Diffie-Hellman
Signing Algorithm	DSA
Encryption Algorithm	RC4 with 128-bit key
Message Digest Algorithm	SHA-1
FIPS	NO
Cipher Suite Name	RC4-SHA
Standard Cipher Suite Name	SSL_RSA_WITH_RC4_128_SHA TLS_RSA_WITH_RC4_128_SHA
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	RC4 with 128-bit key
Message Digest Algorithm	SHA-1
FIPS	NO
Cipher Suite Name	RC4-MD5
Standard Cipher Suite Name	SSL_RSA_WITH_RC4_128_MD5 TLS_RSA_WITH_RC4_128_MD5
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	RC4 with 128-bit key
Message Digest Algorithm	MD5
FIPS	NO
Cipher Suite Name	ADH-AES256-SHA
Standard Cipher Suite Name	TLS_DH_anon_WITH_AES_256_CBC_SHA
Key Exchange Algorithm	Anonymous Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	AES with 256-bit key in CBC mode

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	ADH-AES128-SHA
Standard Cipher Suite Name	TLS_DH_anon_WITH_AES_128_CBC_SHA
Key Exchange Algorithm	Anonymous Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	AES with 128-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	ADH-DES-CBC3-SHA
Standard Cipher Suite Name	SSL_DH_anon_WITH_3DES_EDE_CBC_SHA TLS_DH_anon_WITH_3DES_EDE_CBC_SHA
Key Exchange Algorithm	Anonymous Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	Triple DES (EDE) with 168-bit key in CBC mode
Message Digest Algorithm	SHA-1
FIPS	YES
Cipher Suite Name	ADH-RC4-MD5
Standard Cipher Suite Name	SSL_DH_anon_WITH_RC4_128_MD5 TLS_DH_anon_WITH_RC4_128_MD5
Key Exchange Algorithm	Anonymous Diffie-Hellman
Signing Algorithm	None
Encryption Algorithm	RC4 with 128-bit key
Message Digest Algorithm	MD5
FIPS	NO
Cipher Suite Name	NULL-SHA

Table 12 NSA Suite B Compliant Cipher Suites (defined in RFC 5289)

Standard Cipher Suite Name	SSL_RSA_WITH_NULL_SHA TLS_RSA_WITH_NULL_SHA
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	None
Message Digest Algorithm	SHA-1
FIPS	NO
Cipher Suite Name	NULL-MD5
Standard Cipher Suite Name	SSL_RSA_WITH_NULL_MD5 TLS_RSA_WITH_NULL_MD5
Key Exchange Algorithm	RSA
Signing Algorithm	RSA
Encryption Algorithm	None
Message Digest Algorithm	MD5
FIPS	NO

FIPS 140 compliance

The U.S. government defines the 140 series of Federal Information Processing Standards (FIPS) computer security standards that specify requirements for cryptography modules. ECOM now supports FIPS-compliant encryption algorithms. The following specific changes were made to ECOM for FIPS compliance:

- Digest operations MD2/MD5 were not FIPS compliant and have been replaced by SHA256 (FIPS compliant).
- Cryptography operations previously using RC4 have been replaced by AES (FIPS compliant) in its FIPS-compliant mode. RC4 is still used when ECOM is not run in its FIPS-compliant mode.

Note: Previous adopters of ECOM that have their passphrase.dll and/or role files encrypted with RC4 will have to perform two steps in order to convert their files to AES encrypted files in FIPS-compliant mode:

1. Decrypt the passphrase.dll and/or role files with ECOM's Encrypt tool
2. Re-encrypt passphrase.dll and/or role files using AES with ECOM's Encrypt tool

---

- The EncryptionAlgorithm setting has been deprecated in the security settings file Security_settings.xml and is ignored. ECOM uses AES by default in FIPS-compliant mode and RC4 in non-FIPS-compliant mode.
- A configuration setting in the Security_settings.xml file controls FIPS compliance:

  ```
  <ECOMSetting Name="FIPSMode" Type="boolean"
    Value="true"/>
  ```

  With this setting, ECOM adopters are able to initialize ECOM in FIPS 140 or non-FIPS 140 compliance modes. This is useful to adopters who may want to keep compatibility with legacy applications that are non-FIPS compliant.

The following is a list of concerns that you may face when ECOM runs in the FIPS 140-compliant mode:

PKCS#12 messages

ECOM can import PKCS#12 messages. Ensure that the algorithm used to generate/export a PKCS#12 message is FIPS compliant. For example, Microsoft commonly exports PKCS #12 files using a default encryption algorithm of RC2, which is not FIPS compliant. Also, OpenSSL uses RC2-40 by default to generate PKCS#12 messages, unless the "-descert" option is used (in this case triple DES is used). In addition to the encrypted PKCS #12 message itself, you must be concerned with the algorithms used in the certificate and its key within. The most common reason for problems with the certificates is use of MD5 as part of the signature algorithm.

Certificate and Private Key files

Certificates and private keys that are going to be imported by ECOM must also use FIPS-compliant algorithms if ECOM is run in FIPS-compliant mode.

SSL communication

Only TLS v1 and TLS v1.1 are supported by ECOM in FIPS-compliant mode.

Different library contexts (one in FIPS mode and one in non-FIPS mode)

If an ECOM provider creates a non-FIPS context while ECOM is running in FIPS-compliant mode, ECOM may have a failure related to a known BSAFE issue. BSAFE has global variables to manage FIPS modes, and once a non-FIPS context is created, the whole BSAFE library works in non-FIPS mode only.

Encrypted role files no longer shipped by default

Encrypted role files previously shipped with ECOM are no longer included. ECOM adopters are responsible for generating all the encrypted files they need (by using the Encrypt tool) according to the FIPS mode they are running with ECOM.

The ECOM files now needing encryption depend on your ECOM configuration and may include:

- SecurityPlugin_RolesEncrypted.xml
- DirectoryProvider_RolesEncrypted.xml
- ECOMRolesEncrypted.xml
- PersistenceProvider_RolesEncrypted.xml
- SecurityProvider_RolesEncrypted.xml
- IndicationSubscriptionProvider_RolesEncrypted.xml
- IndicationSubscriptionProvider/IndicationSubscriptionProvider_RolesEncrypted.xml
- RepositoryProvider_RolesEncrypted.xml
- SampleProvider_RolesEncrypted.xml
- ServerProfileProvider_RolesEncrypted.xml
- SLPProvider_RolesEncrypted.xml
- SLPProvider_RolesEncrypted.xml

Using the Encrypt command line tool

The "Encrypt" tool is a command line tool used to generate encrypted role files. By default, it is included in ECOM's bin directory. The command line pattern for usage is:

```
Encrypt <SourceFile> <EncryptedDestinationFile> [-RC4]
```

to encrypt a roles file.

```
Encrypt <EncryptedSourceFile> <DestinationFile>  -decrypt
[-RC4]
```

to decrypt a roles file.

The AES-128-OFB algorithm is used by default. You can specify the "-RC4" option if you want to use RC4 algorithm.

Usage examples

The following Windows examples demonstrate Encrypt usage.

```
<ECOM_install_dir>/bin/Encrypt.exe <ECOM
    folder>/conf/SecurityPlugin_Roles.xml
    <ECOM_install_dir>/conf/SecurityPlugin_RolesEncrypte
    d.xml

<ECOM_install_dir>/bin/Encrypt.exe <ECOM
    folder>/conf/SecurityPlugin_Roles.xml
    <ECOM_install_dir>/conf/SecurityPlugin_RolesEncrypte
    d.xml -RC4
```

SSL Setup for Authentication

In order for SSL communications between two peers to be authenticated, one of the following conditions must exist:

- If a peer presents a self-signed certificate, the host receiving the self-signed certificate must have its trust store seeded with that certificate.
- If a peer presents a CA-signed certificate, the host receiving the CA-signed certificate must have its trust store seeded with a chain of certificates starting from the issuer of the peer's certificate and ending with the root certificate.

Installing certificates in trust stores is performed at configuration time, not at runtime. The following sections describe how to supply certificates to both the ECOM server and its client.

Supplying a client with the ECOM Server Certificate

1. Obtain the ECOM certificate (`ecomtls.crt`) from the directory `[ECOM_Home]\conf\ssl`.
2. If `ecomtls.crt` does not exist, point your browser to the ECOM Admin page `https://<server>:<port>/ECOMConfig`. The connection will fail as the trust store has not been setup yet but the certificate will be generated.
3. Add the ECOM certificate (`ecomtls.crt`) to the client's truststore. The certificate is in PEM format.

Supplying ECOM with the Client Certificate

To authenticate the client certificate, you must import the client certificate into the ECOM truststore. To do this, you must append the certificate to the file `ecomtls.ca` found in the following directory:

```
[ECOM_HOME]\conf\ssl
```

Follow these steps:

1. Obtain the client certificate from an SSL certificate provider.

Note: ECOM accepts certificates in PEM format only at this time.

2. Point your browser to the ECOM Administration Login page:

   ```
   https://{ServerName}:5989/ECOMConfig
   ```

3. Select the **SSL Certificate Management** submenu.

4. Select **Import CA certificate file** to import the certificate. You do this by cut/pasting the certificate to the end of the list of already existing certificates if any exist.
5. Restart ECOM .

Selecting a security plug-in

Authentication and authorization in ECOM are based on a pluggable architecture that allows you to use an existing security plug-in shipped with ECOM or to build a custom security plug-in with the ECOM SDK.

CST Security Plug-in

ECOM includes a security plug-in based on the RSA Common Security Toolkit (CST). "CST Security Plug-in" on page 100 provides an overview of this plug-in. It is also described in greater detail in Appendix C, "CST Security Plug-in."

IMPORTANT

The CST Security Plug-in is used by ECOM by default. Therefore, you must have the RSA Common Security Toolkit installed on the same platform before ECOM startup. By default, ECOM also expects to be able to access a pre-initialized lockbox running in System Mode. You must configure ECOM to access the local CST installation. Appendix C, "CST Security Plug-in" **presents relevant configuration details.**

Custom security plug-ins

Appendix G, "Using the ECOM SDK"provides basic information about using the ECOM SDK to build a custom security plug-in for ECOM. Your compiled plug-in library must be added to your ECOM installation and configured before ECOM startup.

Security plug-in configuration

You can set which security plug-in ECOM loads at startup by changing the PluginName parameter value in the Security_settings.xml configuration file. The default setting:

```
<ECOMSetting Name="PluginName" Type="string"
  Value="CSTSecurityPlugin"/>
```

loads the CST Security Plug-in library CSTSecurityPlugin.dll (libCSTSecurityPlugin.so on Linux). You can specify a different value to enable a custom security plug-in implementation. The compiled library for the configured security plug-in should be placed in the providers directory of your ECOM installation.

IMPORTANT

On startup, if ECOM cannot find and load the security plug-in library specified, it will shut down.

CST Security Plug-in

By default, the CST Security Plug-in is responsible for authentication and role management services in ECOM. The CST Security Plug-in must be properly configured and loaded at ECOM startup for clients to access any data or functionality exposed by providers and OSLS plug-ins.

Authentication

ECOM is preconfigured to load the CST Security Plug-in and use a predefined administrator account in a local, file-based directory for authentication and authorization. The administrator credentials are:

- Username: LocalDir/admin
- Password: #1Password

These default credentials are dynamically generated and placed in the file <ecom_conf_dir>\cst\LocalDirectoryData.xml and RoleData.xml when ECOM first starts. This user account is assigned the administrator role.

Note: You should change these default administrator credentials after first startup. The credentials can be changed using the CST command line tool or through the ECOM Administration Web Server.

The CST Security Plug-in can be configured to use any local and LDAP directories configured and available through a preexisting CST installation. It can also use local and domain users defined on Windows hosts. Appendix C, “CST Security Plug-in” provides configuration instructions for accessing other CST authorities.

Note: The CST forces password expiration after 90 days. Providers detect user password expiry when authenticating and change the expired password using IUserManager interface.

Transmitting client username and password

If the CST Security Plug-in is used by ECOM, usernames may be transmitted in three formats:

- *username* — When only one local CST authentication authority is defined. ECOM assumes that the user should be authenticated against this single local authority. "LocalDir/" does not need to be prepended.

Note: ECOM ships with the default CST Config.xml file, which defines multiple authorities. Therefore, the following username formats are generally necessary.

- *CST_authority_name/username* — When multiple CST authorities are defined.
- *username@CST_authority_name* — When multiple CST authorities are defined.

IMPORTANT

If the name of the authority is not specified and if multiple CST authorities are defined, ECOM will not be able to authenticate the user through CST.

Administering CST user accounts

The ECOM Administration Web Server includes features that allow you to add, change passwords for, and remove user accounts for CST Local Directory Authorities. "Accessing the ECOM Administration Web Server" on page 61 provides more information. The **cstadmin CLI** also allows more extensive user management. The *RSA Common Security Toolkit Developer's Guide* provides more information.

User administration for LDAP directories must be handled by the appropriate external management software.

Limitations

The CST Security Plug-in does not support instance-level authorization.

Authorization for security functions

The file SecurityPlugin_Roles.xml (or SecurityPlugin_RolesEncrypted.xml) defines the security functions supported by the CST Security Plug-in and the corresponding roles required by an authenticated client. You can modify this file to change the required security roles for each function.

4
ECOM Communication Security

ECOM implements security at a variety of levels. With built-in support for the SSL 3.0 and TLS 1.0 protocols, ECOM secures and protects network requests, responses, and indication deliveries. The following sections discuss how to set up and configure ECOM communication security:

ECOM port security

By default, ECOM listens for HTTP(S) requests at four ports defined in Port_settings.xml and specified by the parameters Port0 through Port3, as shown in Figure 24 on page 106. However, ECOM can listen at up to ten ports (Port0 through Port9). Each port can be configured to support secure or unsecure communication. Each port can also be configured to listen for SLP requests or to ignore them. The following XML tags are used to define the configuration of each port:

- <ECOMSettings> — Serves as the root tag of ECOM configuration files.
- <ECOMSetting Name="PortX"> — Defines the port to configure and its name. Valid names are Port0 through Port9. Ports 0 through 3 are configured by default. <port>, <secure>, <slp>, <interfaces>, and <trafficlist> tags are nested within this element.
- <port> — Defines the network device port that ECOM listens on.

Note: To use a port number less than 1024 on a Linux system, ECOM must start with root privileges, but after port binding it does not need to continue as a root user. See "Running ECOM with non-root privileges" on page 59 for more information.

- <portRange> — Allows you to specify a range of ports for ECOM to bind to. The portRange parameter takes a comma separated list of entries. Each entry may be a single number or a range of numbers separated by "-". For example:

  ```
  <portRange>
  1000, 2000-2010, 3000, 4000
  </portRange>
  ```

  This setting tells ECOM to try to bind to port 1000, then to a port in the range 2000 through 2010, then to port 3000, then to port 4000. Port 1000 is attempted first, followed by 2000, 2001, and so on. Once ECOM successfully binds to a port in this range, no other ports in the range are attempted.

  If you use the `portRange` feature, the `ListenerStartTimeout` feature in `ECOM_settings.xml` should be set to a value large enough to accomodate bind attempts on the full set of ports. ECOM development recommends 10 milliseconds per listed port. For example, if you specify a port range of 2000 - 2009, set the `ListenerStartTimeout` to 100.

- <secure> — Defines whether ECOM will use HTTPS (true) or unsecure HTTP (false) on this port. "Securing ECOM communication" on page 109 provides additional information about ECOM and its use of SSL 3.0/TLS 1.0 for secure port communications
- <slp> — Defines whether ECOM will register with the SLP daemon to receive SLP requests on this port (true) or not (false).
- <interfaces> — Defines the IPv4 and IPv6 addresses that ECOM will listen on for a <port>. This configuration tag is optional. By default, ECOM listens on all available IP addresses with the setting 0.0.0.0. <interface> tags are nested within <interfaces>.
- <interface> — Defines one or more specific IP address that ECOM will listen on. If one or more <interface> elements are defined, ECOM will only listen on the defined addresses. <ip> and <ip> tags are nested within an <interface>.
- <ip> and <ip> — Define an IP address used by an interface.
- <trafficlist> — Defines a set of traffic rejection or forwarding rules to apply for this port. This configuration tag is optional. <traffic> tags are nested within a <trafficlist>.
- <traffic> — Defines a specific traffic rule to apply for this port. <reject> and <forward> tags are nested elements for <traffic>. This element requires a type attribute that defines the type of traffic this rule applies to. Possible traffic types are:
  - CIM
  - HTTP_PUT
  - HTTP_GET
  - HTTP_MPOST
  - HTTP_POST
  - ALL_TRAFFIC

  <traffic> elements are processed in the order they are read. Putting a traffic rule of type ALL_TRAFFIC as the first tag, for example, causes all subsequent traffic elements to be ignored.

- <reject> — Specifies that all traffic of the parent type on this port should be ignored. This empty element has no attributes and requires no value.
- <forward> — Specifies the port name (Port0 through Port9) to forward this port's traffic to.
- <allow> — Specifies that all traffic of the parent type should be allowed on this port. This empty element has no attributes and requires no value.

Figure 24, "Sample, custom ECOM port settings defined in Port_settings.xml" demonstrates the syntax used for custom port configuration.

```
<ECOMSetting Name="Port5">
    <port>80</port>
    <secure>false</secure>
    <slp>false</slp>

    <trafficlist>
          <traffic type="HTTP_PUT>
             <reject></reject>
          </traffic>

          <traffic type="ALL_TRAFFIC">
             <forward>Port6</forward>
             </traffic>
    </trafficlist>
</ECOMSetting>

<ECOMSetting Name="Port2">
   <port>8888</port>
   <secure>false</secure>
   <slp>true</slp>
   <interfaces>
      <interface>
         <ip>127.0.0.1</ip>
      </interface>
   </interfaces>
</ECOMSetting>

...
```

Figure 24 Sample, custom ECOM port settings defined in Port_settings.xml

ECOM will successfully startup only if one or more ports are properly defined in Port_settings.xml and ECOM can read the file. If no ports can be read, ECOM will fail to start and log an error, such as:

```
-E- ECOM cannot read any port data entry from
Port_settings.xml. Please check for any errors in this
file. Aborting....
```

portRange Parameter

The portRange parameter allows you to specify a range of ports for ECOM to bind to. For example:

```
<portRange>
1000, 2000-2010, 3000, 4000
</portRange>
```

This setting tells ECOM to try to bind to port 1000, then to a port in the range 2000 through 2010, then to port 3000, then to port 4000. Port 1000 is attempted first, followed by 2000, 2001, and so on. Once ECOM successfully binds to a port in this range, no other ports in the range are attempted.

If you use the portRange feature, the ListenerStartTimeout feature in ECOM_settings.xml should be set to a value large enough to accomodate bind attempts on the full set of ports. ECOM development recommends 10 milliseconds per listed port. For example, if you specify a port range of 2000 - 2009, set the ListenerStartTimeout to 100.

HostName

ECOM supports an optional HostName element in Port_settings.xml. This element is used to specify a fully qualified hostname or IP address for this ECOM instance when advertising its capabilities through SLP.

```
<ECOMSetting Name=”HostName” Type=”string”
Value=”localhost”/>
```

If this value is not found in the file, ECOM dynamically obtains a fully qualified hostname from the operating system.

Note: For SMI-S compliance, ECOM recommends setting HostName to the intended IP address for ECOM or to a fully qualified hostname that resolves accordingly. Do NOT use values of localhost or 127.0.0.1 as these may interfere with the correct operation of any SLP Directory Agents (DA) running on the network.

ECOM also supports an optional HostNameFormat setting.

```
<ECOMSetting Name="HostNameFormat" Type="string"
Value="..."/>
```

This value specifies the format of the hostname that will be advertised for ECOM through SLP. A value of "IP" will advertise the first defined IP address assigned to ECOM. A value of "FQDN" will use the fully qualified domain name defined by the HostName parameter of dynamically obtained from the server.

The ECOM SLP Provider, in combination with the required slpd service agent, listens at port 427 for SLP broadcasts. Appendix A, "SLP Configuration" provides more information about ECOM SLP support.

I.P. filtering

ECOM allows you to limit the I.P. addresses from which it will accept incoming information requests. One or more I.P. addresses can be specified. By default, I.P. filtering is disabled. To enable it, you must:

1. Set the EnableIPFiltering parameter value to true in ECOM_settings.xml.
2. Start or restart ECOM.

I.P. filtering with the ECOM Administration Web Server

To define I.P. filtering with the ECOM Administration Web Server:

1. Log in to the ECOM Administration Web Server. "Accessing the ECOM Administration Web Server" on page 61 provides additional information.
2. From the ECOM Configuration Page, select **IP Filtering**.
3. In the ECOM IP Filtering web page, type a **Trusted client's IP address**.
4. Select **Enabled**.
5. Click **Submit Query**.

Use this same method to add multiple IP addresses.

To disable IP filtering, follow the instructions above, but click **Disable** before clicking **Submit Query**. No IP address needs to be specified.

I.P. filtering with the ECOM SDK

"Using the ECOM SDK" on page 147 presents the ECOM SDK and how to filter I.P. addresses with it.

Securing ECOM communication

For its secured communication ports, ECOM uses SSL 3.0/TLS 1.0 with an RSA 2048-bit encryption algorithm to protect transmissions between itself and clients. An OpenSSL library is included in all ECOM distributions to support this functionality. The ECOM Administration Web Server contains secure communication functionality that allows you to:

- Create a Certificate Signing Request (CSR) for processing by a certificate authority.
- Provide a returned, signed certificate for use by ECOM.

Secure ports

By default, ECOM automatically listens for secure HTTPS requests on port 443 and secure CIM-XML requests on 5989. However, any of the ports ECOM listens on can be configured to be secure in the file Port_settings.xml. "ECOM port security" on page 104 provides more information about ECOM and port security.

Note: To bind an SSL port less than 1024 (such as the standard secure port 443) on a UNIX or Linux system, the ECOM process must run as root.

Configuration settings

ECOM's secure communication capabilities are configured in the file Security_settings.xml. The relevant parameter settings in the file are:

- CertificatesDirectory — Indicates the directory ECOM will look in for the SSL 3.0\TLS 1.0 certificate information used by ECOM for secure communication. The default value is conf/ssl.
- SSLConfigFile— The location of ECOM's default SSL configuration file, named cert_config.cnf by default. This file contains information required by ECOM to generate its own SSL\TLS certificate and private key. Do NOT overwrite or remove this file. Ensure that this file exists in the directory specified in the attribute CertificatesDirectory.
- SSLCertFile — Defines the SSL\TLS certificate file used by ECOM. The default value is ecomtls.crt. Ensure that this file exists in the directory specified in the attribute CertificatesDirectory. You should ensure that the certificate parameter fields in this file reflect your organization and your security requirements.

- SSLUserGeneratedCsrFile — Defines the SSL certificate signing request file created by ECOM when a user enters new certificate information through the ECOM Administration Web Server.
- SSLAutoGeneratedCsrFile — Defines the SSL certificate signing request file created by ECOM automatically at startup if a user-specified file is not configured for use.
- SSLKeyFile— Specified the file that contains the private key used for SSL\TLS. The default value is ecomtls.pk. Ensure that this file exists in the directory specified in the attribute CertificatesDirectory.
- SSLCAFile — Specifies the file that contains the trusted certificate authorities for ECOM, allowing ECOM to check client certificates if configured to do so. This file is empty by default. Content for this file is only needed when SSLClientAuthentication is set to a value of Required or Optional. The default value is ecomtls.ca. Ensure that this file exists in the directory specified in the attribute CertificatesDirectory.
- SendSSLClientCertificate — When ECOM is acting as a CIM client, the server may ask the client to present its certificate during an SSL handshake. According to the defining specification, it is not mandatory for the client to send one. If no certificate is sent, it is up to the server whether to continue the connection or not. When set to true, this configuration setting causes ECOM to send its certificate as the client certificate for the handshake.
- SendSSLClientCertificate — When ECOM is acting as a CIM client, the server may ask the client to present its certificate during an SSL handshake. According to the defining specification, it is not mandatory for the client to send one. If no certificate is sent, it is up to the server whether to continue the connection or not. When set to true, this configuration setting causes ECOM to send its certificate as the client certificate for the handshake.
- SSLClientAuthentication — Indicates the behavior of ECOM with respect to client SSL access. The default value is `Optional`. Possible values are:
  - None — Tells ECOM not to request an SSL certificate from the client.

- Optional — Instructs ECOM to request an SSL certificate from the client but not to require a returned certificate. If no certificate is returned, ECOM permits client access. If a certificate is returned, ECOM will attempt to verify it with its trusted Certificate Authority file (ecomtls.ca). If verification fails, the connection will be terminated and the client will be denied access.
- Required — Instructs ECOM to request an SSL certificate from the client. If no certificate is returned or if ECOM cannot verify the returned certificate with its trusted Certificate Authority file (ecomtls.ca), the connection will be terminated and the client will be denied access.

◆ SSLServerAuthentication — This setting controls whether the server certificate is verified against the ECOM trust store when ECOM acts as a client to deliver indications. The default value of `false` (to keep backward compatible behavior) skips the validation.

◆ `SSLProtocolOptions` - Restricts the allowed SSL protocol versions in both the client and server. More than one option can be disallowed at the same time. List the options as a comma-separated list.

SSLv2 is disabled by default because it contains a number of security flaws.

- `NO_SSLv3` - Do not allow an SSLv3 connection. This option is only applicable when connecting with SSLv23. It prevents the peers choosing SSLv3 as the protocol version.
- `NO_TLSv1` - Do not allow a Transport Layer Security (TLS) 1.0 connection. This option is only applicable when connecting with SSLv23. It prevents the peers choosing TLS 1.0 as the protocol version.
- `NO_TLSv11` - Do not allow a Transport Layer Security (TLS) 1.1 connection. This option is only applicable when connecting with SSLv23. It prevents the peers choosing TLS 1.1 as the protocol version.
- `NO_TLSv12` - Do not allow a Transport Layer Security (TLS) 1.2 connection. This option is only applicable when connecting with SSLv23. It prevents the peers choosing TLS 1.2 as the protocol version.

◆ SSLProtocol - Allows you to configure the level of SSL / TLS your application requires. The settings are as follows:

- `SSLv23` - This is the default. This setting sets up negotiation with the peer.
- `SSLv3` - Dictates the use of the SSL 3.0 protocol.
- `TLSv1` - Dictates the use of the TLS 1.0 protocol.
- `TLSv11` - Dictates the use of the TLS 1.1 protocol.
- `TLSv12` - Dictates the use of the TLS 1.2 protocol.

Using custom certificates with ECOM

If you do not want to use the default SSL certificate generated by ECOM, you can use a different certificate from a third party application or service. The following Wiki page provides sample instructions for generating a custom SSL certificate with OpenSSL and using it in ECOM:

http://cmgwiki.lss.emc.com/index.php/ECOM_HOW-TOs

A custom certificate can be used to:

- Further secure ECOM SSL communication with CIM-XML clients
- Allow ECOM to deliver indications via SSL

Adding the custom certificate to ECOM

To use a custom certificate with ECOM:

1. Obtain SSL/TLS certificate files from a trusted authority. If ECOM will be delivering indications via secure communication, obtain the certificate files from the server receiving the indications. ECOM requires the following SSL files:
   - *.crt (certificate)
   - *.ca (certificate authority)
   - *.pk (private key)
2. Change the following configuration settings in Security_settings.xml to reflect the location and file names of the new certificate.
   - CertificatesDirectory
   - SSLCertFile
   - SSLKeyFile
   - SSLCAFile
3. Start/restart ECOM.

A

SLP Configuration

This section presents detailed information for configuring and deploying the ECOM SLP Provider. This provider is required for SMI-S compliance.

Overview

The ECOM SLP provider allows management applications and services (*user agents*) within a network to find needed instances of ECOM using the Service Location Protocol (SLP). When it receives an SLP multicast indicating a need for its exposed resources, ECOM (a *service agent*) responds to the sender with access details to facilitate direct communication. See Figure 3 on page 18 for an example of an SLP multicast. By default, ECOM listens on port 427 for SLP broadcasts.

Requirements

To respond to SLP requests, the ECOM SLP Provider requires:

- a running instance of an OpenSLP `slpd` service agent (SA) on the same platform as ECOM.

Note: ECOM has only been qualified with OpenSLP 2.0 (alpha) on Windows and 1.2.1 on Linux.

- multicast enabled within its subnet.

IMPORTANT

On Windows platforms, ECOM and the OpenSLP library used by the SLP Provider, require Microsoft's Visual C++ runtime files. See "Windows environments" on page 72 for additional information about obtaining these files.

`slpd`

EMC distributes compiled, binary distributions of OpenSLP in the ECOM Toolkit available at:
http://cmgdistributions.eng.emc.com

After obtaining the OpenSLP binaries included with the ECOM Toolkit, follow the instructions below to run it.

Note: You must have the ECOM SLP Provider configured on a running ECOM instance to receive an SLP response from ECOM. You must also have the `slpd` service agent running on the same platform before ECOM startup to ensure that ECOM can listen for SLP multicast requests. `slpd` can also be started after ECOM startup, however ECOM may not detect it quickly depending on its configuration. See Appendix A, "SLP Configuration" for additional information.

EMC recommends that you start `slpd` as a Windows service or Unix daemon before ECOM startup. The ECOM SLP Provider looks for a running `slpd` service agent at startup. If found, ECOM will use the running instance. If no service agent is found, ECOM will periodically check for it.

Note: See "Configuration" on page 118 for information on how to set the ECOM SLP Provider's polling interval.

Starting slpd on a Windows host

The following example presents instructions for starting `slpd` as a service on a Windows-based platform. Startup instructions for other platforms will vary. To start `slpd`:

1. Open a system command line and navigate to the location of the `slpd.exe` and `slptool.exe` binaries.

2. If you have not done so before, install the `slpd` service agent as a Windows service with the command:

   ```
   slpd.exe -install auto
   ```

   If you look in the Windows Services dialog box, the listed name for the service is "Service Location Protocol." This service is not started at installation.

3. Ensure that the two configuration files for `slpd` are installed on the platform. Place them in the same directory as `slpd.exe` or in a different location. The following steps refer to the location of these configuration files as `<slp_conf_dir>`. The configuration files used by `slpd.exe` are:

   - `slp.conf` (Required)—A configuration file for `slpd.exe` that contains information affecting slpd's operation as either a directory agent, service agent, or both
   - `slp.reg` (Optional)—A legacy file that does not require modification

4. To specify the IP address that `slpd` listens for SLP broadcasts on, modify the attribute `net.slp.interfaces` in `slp.conf` to reflect the IP address of ECOM on your platform. You may need to remove the preceding semicolon to ensure that this value is not commented out. If you do not specify this value, `slpd` will automatically select an IP address to use at startup. Multiple IP addresses to listen at can be specified by separating them with commas. Do not leave whitespaces in between the commas and addresses.

Note: EMC highly recommends setting this value manually to ensure the correct IP address is found and used.

5. Start `slpd` as a Windows service from the Windows **Services** dialog box, which is available from the **Administrative Tools** folder of the **Control Panel**. In the **Services** dialog box:
   1. Right-click **Service Location Protocol** from the list of services and select **Properties**.
   2. In the **Start parameters** field of the **Service Location Protocol Properties** dialog box, enter the following:
      ```
      -c <slp_conf_dir>\slp.conf -r
      <slp_conf_dir>\slp.reg -l <slp_conf_dir>\slp.log
      ```
      substituting the full path to the files on your system in place of `<slp_conf_dir>`. Click **OK** to set the parameters. Note that these parameters will have to be set every time you start the service. Windows does not remember them.
   3. Right-click **Service Location Protocol** from the list of services and select **Start**.
6. Verify that the service agent is running by issuing the following from a command line in your OpenSLP installation directory:
   ```
   slptool.exe findsrvs service:service-agent
   ```

If `slpd` started correctly, you will receive a response similar to:

```
service:service-agent://<host_ip_address>,65535
```

where `<host_ip_address>` is the IP address used by the service agent.

Starting slpd on a Linux host

The following example presents instructions for starting `slpd` on a Linux-based platform. Startup instructions for other platforms will vary. To start `slpd`:

1. Open a system command line and navigate to the location of the `slpd` and `slptool` binaries.
2. Ensure that the two configuration files for `slpd` are installed on the platform. You can place them in the same directory as `slpd` or in a different location. The following steps refer to the location of these configuration files as `<slp_conf_dir>`. The configuration files used by `slpd` are:

- `slp.conf` (Required)—A configuration file for `slpd` that contains information affecting slpd's operation as either a directory agent, service agent, or both
- `slp.reg` (Optional)—A legacy file that does not require modification

3. Modify the attribute `net.slp.interfaces` in `slp.conf` to reflect the IP address of ECOM on your platform. You may need to remove the preceding semicolon to ensure that this value is not commented out. Multiple IP addresses can be specified if separated with commas. Do not leave whitespaces in between the commas and addresses.

Note: EMC highly recommends setting this value manually to ensure the correct IP address is found and used.

4. Assuming the SLP configuration files are placed in the same directory as the executable `slpd`, start `slpd` as a Linux daemon with the command:
   ./slpd -c slp.conf -l slp.log -r slp.reg

IMPORTANT

`slpd` on Linux must be run with `root` permissions.

5. Verify that the service agent is running by issuing the following from a command line in your OpenSLP installation directory:
   ```
   ./slptool findsrvs service:service-agent
   ```

If `slpd` has started correctly, you will receive a response similar to:

```
service:service-agent://<host_ip_address>,65535
```

where `<host_ip_address>` is the IP address used by the service agent.

Note: Additional information about OpenSLP installation on Linux can be found on the OpenSlp at:
http://www.openslp.org/doc/html/UsersGuide/Installation.html

multicast

Multicast within a local subnet is required to allow a user agent to widely broadcast a need for a managed resource to all available service agents in a network. On many networks and networking devices, multicast is blocked intentionally or by default. To allow user agents to send multicast requests to ECOM, you must enable multicast in at least the subnet in which user agents and service agents reside, including ECOM. Consult your network administrator for additional information and assistance.

Configuration

Follow these steps to ensure that the ECOM SLP Provider is configured correctly to run in ECOM.

1. Ensure that the general configuration file for the ECOM SLP Provider `SLPReg_settings.xml` is properly placed in `<ECOM_install_dir>\conf`. See "Understanding ECOM configuration files" on page 80 for additional information about this location. See Figure 26 on page 86 for an example of the contents of this file. This file contains two needed parameters:

   - `interval`—Indicates how often the ECOM SLP provider will refresh its SLP service template registration. This value is specified in minutes.
   - `retrytimes`—Indicates how many times the ECOM SLP Provider will retry a service template registration if its first attempt failed. If all attempts fail, it will wait `interval` minutes before trying again.

2. Ensure that the ECOM SLP Provider library (`SLPProvider.dll` for Windows), the provider registration file `SLPProvider_prg.xml` (Figure 25), and the provider's role security file `SLPProvider_Roles.xml` (Figure 26) exist in the appropriate directories specified in `PluginManager_settings.xml`. "Understanding ECOM configuration files" on page 80 provides additional information about the placement of these files. `SLPProvider_prg.xml` does not require any additional configuration changes, and EMC does not recommend modifying these default values.

```
<?xml version="1.0"?>
<NaviProvider>
      <Namespace Value="root/emc/ecom" />
      <Name Value="SLPProvider" />
      <ProtocolInterface Value="dll" />
      <ApiMapping Value="Navisphere C++" />
      <Vendor Value="EMC Corporation" />
      <Location Value="SLPProvider.dll" />
      <ClassList>
      <Class Name="NaviSLP_Reg" />
      </ClassList>
</NaviProvider>
```

Figure 25 Configuration settings in **`SLPProvider_prg.xml`**

```
<Authorization>
    <Object Name="NaviSLP_Reg">
      <Role Name="Administrator"/>
      <Role Name="Manager"/>
      <Role Name="Monitor"/>
      <Role Name="Provider"/>
    </Object>
</Authorization>
```

Figure 26 Applying ECOM's role based security to the SLP Provider in **`SLPProvider_Roles.xml`**

3. Optionally, you can set a value for the `HostName` attribute of Port_settings.xml (the default is `localhost`), which is a user-defined string, to specify a name or IP address for this ECOM instance when advertising its capabilities through SLP. If this value is not found, ECOM obtains a hostname from the operating system.

Testing response

To test for proper response from a running ECOM SLP Provider, you can use the `slptool` software application included with OpenSLP. The following steps demonstrate how to use this tool on a Windows platform. The instructions for other platforms will vary.

1. Verify that the service agent is running with the command
   ```
   slptool.exe findsrvs service:service-agent
   ```
2. Follow the procedures documented in "Starting and stopping ECOM" on page 58 to start an instance of ECOM.
3. Check ECOM SLP response using the command
   ```
   slptool.exe findsrvs
   service:wbem:https:<your_ecom_ip_address>:5989
   ```

   If ECOM is successfully listening for SLP multicast requests, `slptool` will generate a response similar to that shown in Figure 27.

```
slptool findattrs service:wbem:https://172.23.150.136:5989
(template-type=wbem),(template-version=1.0),(template-description=This
     template describes the attributes used for advertising WBEM
     Servers.),(template-url-syntax=https://172.23.150.136:5989),(service-hi
     -name=CIM Server),(service-hi-description=Device Manager
     CIMOM),(service-id=service:wbem:https://172.23.150.136:5989),(Communica
     tionMechanism=cim-xml),(InteropSchemaNamespace=root/hitachi/dm51),(Prot
     ocolVersion=1.1),(FunctionalProfilesSupported=Basic Read,Basic
     Write,Instance Manipulation,Association Traversal,Query
     Execution,Indications),(MultipleOperationsSupported=false),(Authenticat
     ionMechanismsSupported=Basic),(Namespace=root/hitachi/dm51),(Classinfo=
     CIM
     2.11),(RegisteredProfilesSupported=SNIA:Array,SNIA:Array:Software,SNIA:
     Array:Access Points,SNIA:Array:Disk Drive Lite,SNIA:Array:Extent
     Composition,SNIA:Array:Job Control,SNIA:Array:Masking and
     Mapping,SNIA:Array:FC Target Ports,SNIA:Array:Multiple Computer
     System,SNIA:Array:Location,SNIA:Server,SNIA:Server:Indication)
```

Figure 27 Sample SLP response from a properly-configured ECOM running the `slpd` service agent

B

ECOM Server Profile Provider

This section presents detailed information about the ECOM Server Profile Provider. This provider is required for SMI-S compliance.

Overview

The ECOM Server Profile Provider provides information about ECOM and its exposed resources to management application clients.

A client connecting to ECOM can use the SMI-S Server Profile Provider to:

- Identify the profiles registered with an instance of ECOM.
- Retrieve the entry point for managing a supported profile implementation.
- Identify the SNIA version of a supported profile.
- Determine any subprofiles supported by a profile.
- Retrieve the namespaces exposed by an ECOM instance.
- Retrieve information about the ECOM server itself, such as its version number and operational status.
- Determine the WBEM operations supported by the ECOM instance.
- Determine additional information about ECOM's support for the CIM-XML protocol.

In essence, the Server Profile Provider allows clients connecting to an ECOM instance to know what it is capable of and how to access those capabilities. Additional information for the SMI-S Server Registration Profile, including the CIM classes that represent it, can be found within the specification itself:

http://172.23.146.185/cimom/docs/smis1.2/CommonProfiles.book.pdf (Internal EMC only)

Requirements

The ECOM Server Profile Provider has no third-party dependencies or requirements.

Configuration

Follow these steps to ensure that the ECOM Server Profile Provider is configured correctly to run in ECOM.

- Ensure that the proper version of the SMI-S Server Profile is set in the SMIServerProfileVersion parameter in ECOM_settings.xml.
- Check to ensure that the ECOM Server Profile Provider library (`ServerprofileProvider.dll` for Windows) and registration file `ServerProfileProvider_prg.xml` exist in the appropriate directories specified in `PluginManager_settings.xml`. See "Understanding ECOM configuration files" on page 80 for additional information about the placement of these files.

`ServerProfileProvider_prg.xml` (Figure 28) does not need any additional configuration changes, and EMC does not recommend modifying its default values.

```
<?xml version="1.0"?>
<NaviProvider>
  <Namespace Value="interop" />
  <Name Value="ServerProfileProvider" />
  <ProtocolInterface Value="dll" />
  <ApiMapping Value="Navisphere C++" />
  <Vendor Value="EMC Corporation" />
  <Location Value="ServerProfileProvider.dll" />
  <ClassList>
     <Class Name="CIM_ManagedElement">
        <Qualifier name="Abstract" />
        <Qualifier name="Version">
           <Value>2.10.0</Value>
        </Qualifier>
        <Qualifier name="Description">
           <Value>ManagedElement is an abstract class that provides a common
           superclass (or top of the inheritance tree) for the
           non-association classes in the CIM Schema.</Value>
        </Qualifier>
        <Qualifier name="Dynamic" />
        <Qualifier name="Provider">
           <Value>ServerProfileProvider</Value>
        </Qualifier>
        <Qualifier name="Roles">
           <Value>administrator</Value>
           <Value>manager</Value>
           <Value>monitor</Value>
           <Value>provider</Value>
        </Qualifier>
        <Property Name="Caption" Type="string">
             <Qualifier name="Description">
             <Value>The Caption property is a short textual description (one-
               line string) of the object.</Value>
          </Qualifier>
          <Qualifier name="MaxLen">
             <Value>64</Value>
          </Qualifier>
        </Property>
...
```

Figure 28 Configuration settings in `ServerProfileProvider_prg.xml`

Polling plug-ins

To determine if any new ManagedElement has been instantiated after startup so that it can create required ElementConformsToProfile associations for a plug-in, ECOM polls each plug-in at a pre-determined interval that is configurable. Optionally, polling can be shut off entirely. The relevant parameter is ServerProfilePollInterval in ServerProfile_settings.xml. The default is 60 seconds. A value of 0 turns off polling.

Testing response

Use the OpenPegasus client to test the response of ECOM through the Server Profile Provider. The OpenPegasus command line client can be downloaded from:

http://www.openpegasus.org/

The following is a sample command line for use with a running version of ECOM to collect information from ECOM using the `EnumerateInstances` CIM operation. The response is shown in Figure 29.

```
C:\pegasus\pegasus_bin\bin>cli -l localhost:5988 -n
root/emc ei CIM_System
```

```
path= ECOM_System.CreationClassName="ECOM_ComputerSystem",
      Name="MyHost.eng.emc.com"
//Instance of Class ECOM_System
instance of class ECOM_System
{
string CreationClassName = "ECOM_ComputerSystem";
string Name = "USENSDOGRAL2E.eng.emc.com";
string NameFormat = "Other";
string PrimaryOwnerName = "";
string PrimaryOwnerContact = "";
string Roles[] = ;
sint32 EnabledState = 0;
string OtherEnabledState = "";
sint32 RequestedState = 0;
sint32 EnabledDefault = 0;
uint16 OperationalStatus[] = ;
string StatusDescriptions[] = ;
string Caption = "Computer System";
string Description = "Computer System";
string ElementName = "";
};
```

Figure 29 A sample response from ECOM's Server Profile Provider obtained using the OpenPegasus client to send an `EnumerateInstances` CIM operation request on objects of class type `CIM_System`

C

ECOM Indication Subscription Provider

This section presents detailed information about the ECOM Indication Subscription Provider. This provider is required for SMI-S compliance.

Overview

The ECOM Indication Subscription Provider allows ECOM to support the CIM Event Model and allows clients to create indication subscriptions to receive notification of events of interest in underlying managed resources. Specifically, three CIM objects are managed by this provider:

- EMC_IndicationFilter inheriting from CIM_IndicationFilter
- EMC_ListenerDestinationCIMXML inheriting from CIM_ListenerDestinationCIMXML
- EMC_IndicationSubscription inheriting from CIM_IndicationSubscription

For an overview of the CIM Event Model, see:

http://wbemsolutions.com/tutorials/CIM/cim-model-event.html

A more detailed white paper entitled the "CIM Event Model White Paper" is available at:

http://www.dmtf.org/standards/documents/CIM/DSP0107.pdf

An UML overview of the CIM Event Schema is available at:

http://www.dmtf.org/standards/cim/cim_schema_v214/CIM_Eve nt.pdf

Specific information about ECOM's support for CQL and indications can be found in Appendix E, "CQL Support" and Appendix F, "CIM Indication Support".

This provider is preconfigured to operate correctly within ECOM. You should not move or modify its files unless you wish to remove ECOM's support for indications entirely.

Note: ECOM can send indications using both HTTP (non-secured) and HTTPS/SSL (secured).

Creating persistent connections for indication delivery

CIM clients connecting to ECOM can create a persistent connection using the CreateInstance intrinsic method to create an instance of EMC_ListenerDestinationCIMXML (as part of the broader subscription process) that must have the property PertistentType set to transient (value "3") and the property isSessionBased set to "true". Once this instance is created, ECOM leaves open the connection over which the request came in. This connection is then used to deliver indications back to the client/listener. Figure 30 on page 127 provides an example of a persistent connection request.

```
POST /cimom HTTP/1.1
HOST: USENSANEPD1C.corp.emc.com:5988
Content-Type: application/xml; charset="utf-8"
content-length: 0000000758
TE: chunked, trailers
CIMOperation: MethodCall
CIMMethod: CreateInstance
CIMObject: INTEROP
Authorization: Basic YWRtaW46IzFQYXNzd29yZA==

<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="1095" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateInstance">
      <LOCALNAMESPACEPATH>
        <NAMESPACE NAME="INTEROP"/>
      </LOCALNAMESPACEPATH>
      <IPARAMVALUE NAME="NewInstance">
        <INSTANCE  CLASSNAME="CIM_ListenerDestinationCIMXML" >
          <PROPERTY NAME="Name"  TYPE="string">
            <VALUE>PersistentConnection</VALUE>
          </PROPERTY>
          <PROPERTY NAME="Destination"  TYPE="string">
            <VALUE>SAMPLE_VALUE</VALUE>
          </PROPERTY>
          <PROPERTY NAME="PersistenceType"  TYPE="uin16">
            <VALUE>3</VALUE>
          </PROPERTY>
          <PROPERTY NAME="isSessionBased"  TYPE="boolean">
            <VALUE>true</VALUE>
          </PROPERTY>
          </INSTANCE>
      </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Figure 30 Request for a persistent connection for indication delivery

Note: Once the CreateInstance intrinsic method has successfully completed, the initiating CIM client may not send any other requests over this connection, and this connection is used for indication delivery only.

This functionality specifically supports clients running on hosts behind a firewall and clients on hosts with non-static IP addresses.

Requirements

The ECOM Indication Subscription Provider has no third-party dependencies or requirements.

Configuration

The ECOM Indication Subscription Provider is preconfigured to operate correctly within ECOM. However, follow these steps to ensure that required files are present and correctly placed in the event of changes.

- Check to ensure that the ECOM Indication Subscription Provider library (`IndicationSubscriptionProvider.dll` for Windows) and registration file `IndicationSubscriptionProvider_prg.xml` exist in the appropriate directories specified in `PluginManager_settings.xml`. See "Understanding ECOM configuration files" on page 80 for additional information about the placement of these files.

`IndicationSubscriptionProvider_prg.xml` does not need any additional configuration changes, and EMC does not recommend modifying its default values.

Testing response

The successful creation of CIM indication subscriptions demonstrates proper operation for this provider. See Appendix F, "CIM Indication Support" for the steps involved in creating, modifying, and deleting an indication subscription.

D

ECOM Repository Provider

This section presents detailed information about the ECOM Repository Provider. This provider is required for SMI-S compliance.

Overview

The ECOM Repository Provider persists CIM indication subscriptions as XML files in the file system that ECOM is running on. Persisting these subscriptions ensures that clients can continue to receive important information about managed resources after an ECOM restart.

Specific information about ECOM's support for CQL and indications can be found in Appendix E, "CQL Support" and Appendix F, "CIM Indication Support".

This provider is preconfigured to operate correctly within ECOM. You should not move or modify its files unless you wish to remove ECOM's support for indication subscriptions entirely.

Requirements

The ECOM Repository Provider has no third-party dependencies.

Configuration

The ECOM Repository Provider is preconfigured to operate correctly within ECOM. However, you can follow these steps to ensure that required files are present and correctly placed in the event of changes.

- Check to ensure that the ECOM Repository Provider library (`RepositoryProvider.dll` for Windows), configuration file `Repository_settings.xml`, and registration file `RepositoryProvider_prg.xml` exist in the appropriate directories specified in `PluginManager_settings.xml`. See "Understanding ECOM configuration files" on page 80 for additional information about the placement of these files.

`RepositoryProvider_prg.xml` does not need any additional configuration changes, and EMC does not recommend modifying its default values.

You can change the directory that indication subscriptions are saved to using the `RepositoryDirPath` parameter of `Repository_settings.xml` (Figure 31). The specified value is a path relative to the installation directory of ECOM. You can also set the maximum number of indication subscriptions that can be saved using the parameter `MaxInstanceCount`.

Note: CMG supports up to 1000 persisted indication subscriptions with ECOM. Larger values may work well, but they are not supported.

```
<ECOMSettings>
  <ECOMSetting Name="RepositoryDirPath" Type="string"
   Value="local/repos/ecom/instances"/>
  <ECOMSetting Name="MaxInstanceCount" Type="uint32" Value="1000"/>
</ECOMSettings>
```

Figure 31 Sample contents of `Repository_settings.xml`

Testing response

The successful creation of CIM indication subscriptions and their persistence over an ECOM restart demonstrates proper operation for this provider. See Appendix F, "CIM Indication Support" for the steps involved in creating, modifying, and deleting an indication subscription.

E

CQL Support

ECOM’s limited support for the CIM Query Language (CQL) is intended only for creating, modifying, and deleting CIM Indication filters, which are needed for indication subscriptions. Appendix F, “CIM Indication Support” provides more information about ECOM support for CIM indications. The CIM Query Language is defined in DSP202.

Supported CQL

ECOM supports the following features of CQL.

RESERVED and Supported Keywords

- AND
- ANY
- BY
- EVERY
- FALSE
- FIRST
- FROM
- ISA
- NOT
- OR
- ORDER
- SELECT
- SourceInstance
- TRUE
- WHERE (WHERE is optional in a query)

Property Names

- `property-scope`

Expressions

- `expr-term AND expr-factor`
- `expr-term OR expr-factor`
- `array-comp`
- `arith comp-op arith`
- `chain comp-op value-symbol`. *This support is for embedded objects only.*
- `value-symbol comp-op chain`. *This support is for embedded objects only.*
- `arith ISA identifier`
- `comp-op = "=" | "<>" | "<" | "<=" | ">" | ">="`
- `chain "." [property-scope] identifier`. *This support is for embedded objects only.*
- `chain "[" array-index-list "]"`. *Note: ".."* `array-range` *is not supported.*

- ORDER BY clause

Functions

- AVG
- COUNT
- MAX
- MIN
- OBJECTPATH
- SUM

```
SELECT SUM(CIM_RegisteredProfile.RegisteredOrganization),
     COUNT(CIM_RegisteredProfile.RegisteredVersion) FROM
     CIM_RegisteredProfile
```

Figure 32 Example CQL query using functions

Select List

- `star-expr`. *Note:* `classname.*` *is not supported. The use of an alias is also not supported.*

From Criteria

- `from-specifier` *but only one of these specifiers is allowed.*

Class Paths

- `class-path`, but *[literal-string "."] is not supported.*

Array Comparison

Array comparison is supported. For example:

```
SELECT * FROM CIM_CIMXMLCommunicationMechanism WHERE
CIM_CIMXMLCommunicationMechansim.FunctionalProfilesSuppo
rted[4] = 9
```

Examples

The following are example CQL statements supported by ECOM:

```
SELECT * FROM Cim_InstModification where SourceInstance
ISA FC_Port AND previousInstance.started = 7
```

```
SELECT a.b FROM a where a.SourceInstance ISA FC_Port AND
a.previousInstance::started = 7
```

```
SELECT * FROM CIM_ManagedElement WHERE SourceInstance ISA
CIM_RegisteredProfile
```

Known limitations

The following are known limitations for the supported keywords and functions.

- ECOM does not support querying a property without class scope in a SELECT clause. For example, the following statement is not supported by ECOM:

  ```
  SELECT b FROM a where a.SourceInstance ISA FC_Port AND
  a.previousInstance.started = 7
  ```

  There is one exception to this rule. In a WHERE clause, when ISA is used, a property can be specified without class scope. The following example is a valid statement supported by ECOM.

  ```
  SELECT * FROM Cim_InstModification WHERE
  SourceInstance ISA FC_PORT
  ```

CIM Indication Support

ECOM supports the creation, modification, and deletion of CIM indication subscriptions that allow you to receive notification of events of interest pertaining to your managed resources. For an overview of the CIM Event Model used by ECOM, see:

http://wbemsolutions.com/tutorials/CIM/cim-model-event.html

A more detailed white paper entitled the "CIM Event Model White Paper" can be found at:

http://www.dmtf.org/standards/documents/CIM/DSP0107.pdf

An UML overview of the CIM Event Schema can also be found at:

http://www.dmtf.org/standards/cim/cim_schema_v214/CIM_Eve
nt.pdf

The following sections describe the basic indication support included in ECOM, provide CIM-XML examples, and discuss the limitations of this support.

Configuring indications support

ECOM allows you to configure how it supports indications using values defined in Indication_settings.xml, which is placed by default in `<ecom_conf_dir>`.

Indication support threads

The number of threads used by ECOM to deliver indications to subscribing clients can be configured. To set this value, edit the element named `DeliveryThreadCount`. The default value is 5.

Indication delivery ports

The individual ports or port ranges used by ECOM for indication delivery can be specified in the parameter `OutboundConnectionPorts`. This option takes values fitting the POSIX regular expression:

```
^([0-9]{1,5})([-][0-9]{1,5})?(([,][0-9]{1,5})([-][0-9]{1
,5})?)*$
```

Defined local ports are separated by a comma (","). Ranges of ports arespecified by an hyphen "-".

For example:

```
    <ECOMSetting Name="OutboundConnectionPorts"
     Type="string"
     Value="4000-6000,6010,7014,8000-9000,10002"/>
```

This example configuration defines a sequence of port ranges (4000-6000 and 8000-9000) and individual ports (6010, 7014 and 10002) for outbound connections.

Once the list of ports is loaded, ECOM will try to use them by calling bind() as a client socket. The list of ports is implemented as a circular list, and a new port is used for each new indication that is going to be delivered, returning to the list's beginning when the end of the list is reached. If a port is not available (e.g. in use by another process), ECOM will try the next one in the list until all ports have been attempted. If none of the defined ports is available, ECOM will return an error.

If the `OutboundConnectionPorts` parameter is not present, invalid or empty, ECOM will discard the configuration and allow the operating system to choose the ports to use.

Indication Subprofile configuration

You can configure which profiles are associated with the SMI-S Indication Subprofile using an XML element named SubProfileRequiresProfile.

```
<ECOMSetting Name="SubprofileRequiresProfile"
Type="string" Value="Array, HBA"/>
```

The Value attribute for this XML configuration element contains a comma-separated list of profiles to associate with the Indication Subprofile. In the example above, the Array and HBA profiles are associated with the subprofile. If a listed profile name is invalid or does not exist, it will be ignored. Several profiles are linked by default and should not be listed. These include:

- Server
- SMI-S
- Profile Registration

Managing indication subscriptions

ECOM allows clients to manage indication subscriptions through CIM-XML requests. Indication subscriptions are created a by linking a filter (query) defining what is to be monitored to a destination (URL) defining where notifications are to be sent. The following steps demonstrate the lifecycle of creating, modifying, and deleting an indication subscription.

Note: Note: G ????Indication subscriptions must be managed in the `interop` namespace.

1. Define a destination (an object derived from the `CIM_ListenerDestinationCIMXML` class, as a shown in Figure 33) that specifies the client URL to deliver event notifications to. Notifications are delivered through CIM-XML as instances of the `CIM_Indication` class. Notifications must be URLs of the format `http://` or `https://`. If no '`//`' can be found in the defined `Destination` value, the value is pre-pended with '`http://`'. If no port number can be found in the URL, a value of port 80 is assumed.

   The `PersistenceType` value of an indication destination defines whether the indication subscription it is associated with will be persisted across ECOM restarts ('Permanent' with a value of 2) or lost on ECOM restart ('Transient' with a value of 3). Transient destinations and their associated subscriptions reside in memory only and are lost on ECOM restart. Additionally, according to SMI-S requirements, if the indication delivery fails 3 times for a transient destination, the destination will be deleted as well as any subscription instances that refer to that destination.

Permanent destinations and their associated subscriptions are saved to the file system by the ECOM Repository Provider and reloaded into memory when ECOM restarts. If after three attempts an indication cannot be delivered to a permanent destination, ECOM will not attempt further deliveries of the indication. However, the delivery of subsequent indications will be attempted.

```
<?xml version="1.0" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0" >
  <MESSAGE ID="877" PROTOCOLVERSION="1.0" >
   <SIMPLEREQ>
     <IMETHODCALL NAME="CreateInstance" >
       <LOCALNAMESPACEPATH>
       <NAMESPACE NAME="interop" />
       </LOCALNAMESPACEPATH>
       <IPARAMVALUE NAME="NewInstance" >
       <INSTANCE CLASSNAME="CIM_ListenerDestinationCIMXML">
       <PROPERTY NAME="SystemCreationClassName"
         TYPE="string"><VALUE>CIM_System</VALUE>
       </PROPERTY>
       <PROPERTY NAME="SystemName" TYPE="string"><VALUE>thesystem</VALUE>
       </PROPERTY>
       <PROPERTY NAME="CreationClassName"
         TYPE="string"><VALUE>CIM_ListenerDestinationCIMXML</VALUE>
       </PROPERTY>
       <PROPERTY NAME="Name" TYPE="string"><VALUE>test1</VALUE>
       </PROPERTY>
       <PROPERTY NAME="Destination"
         TYPE="string"><VALUE>http://bogus.com</VALUE>
       </PROPERTY>
       <PROPERTY NAME="PersistenceType" TYPE="uint16"><VALUE>2</VALUE>
       </PROPERTY>
       <PROPERTY NAME="ElementName" TYPE="string"><VALUE>User-friendly
         Destination Name</VALUE>
       </PROPERTY>
       </INSTANCE>
       </IPARAMVALUE>
     </IMETHODCALL>
   </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Figure 33 A CIM-XML request to create a destination for a CIM indication subscription

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="877" PROTOCOLVERSION="1.0">
    <SIMPLERSP>
      <IMETHODRESPONSE NAME="CreateInstance">
       <IRETURNVALUE>
         <INSTANCENAME CLASSNAME="EMC_ListenerDestinationCIMXML">
           <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
            VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
           <KEYBINDING NAME="SystemName"><KEYVALUE
            VALUETYPE="string">thesystem</KEYVALUE></KEYBINDING>
           <KEYBINDING NAME="CreationClassName"><KEYVALUE
            VALUETYPE="string">EMC_ListenerDestinationCIMXML</KEYVALUE></KEY
            BINDING>
           <KEYBINDING NAME="Name"><KEYVALUE
            VALUETYPE="string">test1</KEYVALUE></KEYBINDING>
         </INSTANCENAME>
       </IRETURNVALUE>
       </IMETHODRESPONSE>
    </SIMPLERSP>
  </MESSAGE>
 </CIM>
```

Figure 34 **A CIM-XML response to the successful creation of an indication subscription destination**

2. Define a filter (an object of type `CIM_IndicationFilter`, as shown in Figure 35) that defines the trigger for notifications to be sent. Filters are defined using CQL. See Appendix E, "CQL Support" for more information about ECOM's support for CQL.

```
<?xml version="1.0" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0" >
  <MESSAGE ID="877" PROTOCOLVERSION="1.0" >
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateInstance" >
        <LOCALNAMESPACEPATH>
        <NAMESPACE NAME="interop" />
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="NewInstance" >
        <INSTANCE CLASSNAME="CIM_IndicationFilter">
        <PROPERTY NAME="SystemCreationClassName"
          TYPE="string"><VALUE>CIM_System</VALUE>
        </PROPERTY>
        <PROPERTY NAME="SystemName" TYPE="string"><VALUE>systemname</VALUE>
        </PROPERTY>
        <PROPERTY NAME="CreationClassName"
          TYPE="string"><VALUE>CIM_IndicationFilter</VALUE>
        </PROPERTY>
        <PROPERTY NAME="Name" TYPE="string"><VALUE>Filter1</VALUE>
        </PROPERTY>
        <PROPERTY NAME="Query" TYPE="string"><VALUE>SELECT * FROM
          CIM_InstCreation</VALUE></PROPERTY>
        <PROPERTY NAME="QueryLanguage"
          TYPE="string"><VALUE>DMTF:CQL</VALUE></PROPERTY>
        <PROPERTY NAME="ElementName"
          TYPE="string"><VALUE>TestFilter</VALUE></PROPERTY>
        <PROPERTY NAME="SourceNamespace"
          TYPE="string"><VALUE>interop</VALUE></PROPERTY>
        </INSTANCE>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Figure 35 **A CQL statement is placed within a `CIM_IndicationFilter` object sent to ECOM in a CIM-XML request**

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="877" PROTOCOLVERSION="1.0">
    <SIMPLERSP>
      <IMETHODRESPONSE NAME="CreateInstance">
        <IRETURNVALUE>
          <INSTANCENAME CLASSNAME="EMC_IndicationFilter">
            <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
              VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="SystemName"><KEYVALUE
              VALUETYPE="string">systemname</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="CreationClassName"><KEYVALUE
              VALUETYPE="string">EMC_IndicationFilter</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="Name"><KEYVALUE
              VALUETYPE="string">Filter1</KEYVALUE></KEYBINDING>
          </INSTANCENAME>
        </IRETURNVALUE>
      </IMETHODRESPONSE>
    </SIMPLERSP>
  </MESSAGE>
</CIM>
```

Figure 36 A CIM-XML response to the successful creation of an indication filter

3. Create an indication subscription (an object of type `CIM_IndicationSubscription`, as shown inFigure 37) using the handler (`test1`) and filter(`Filter1`) previously defined.

```
<?xml version="1.0" encoding="UTF-8"?>
<CIM CIMVERSION="2.3" DTDVERSION="2.2">
  <MESSAGE ID="2007:2:22:8:0:42:26:10" PROTOCOLVERSION="1.0">
    <SIMPLEREQ>
      <IMETHODCALL NAME="CreateInstance">
        <LOCALNAMESPACEPATH>
        <NAMESPACE NAME="root"/>
        <NAMESPACE NAME="interop"/>
        </LOCALNAMESPACEPATH>
        <IPARAMVALUE NAME="NewInstance">
        <INSTANCE CLASSNAME="CIM_IndicationSubscription">
        <PROPERTY.REFERENCE NAME="Filter" PROPAGATED="false"
          REFERENCECLASS="CIM_IndicationFilter">
        <VALUE.REFERENCE>
        <INSTANCEPATH>
        <NAMESPACEPATH><HOST>127.0.0.1</HOST>
        <LOCALNAMESPACEPATH><NAMESPACE NAME="root"/><NAMESPACE
          NAME="interop"/></LOCALNAMESPACEPATH>
        </NAMESPACEPATH>
        <INSTANCENAME CLASSNAME="EMC_IndicationFilter">
        <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
          VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
        <KEYBINDING NAME="SystemName"><KEYVALUE
          VALUETYPE="string">systemname</KEYVALUE></KEYBINDING>
        <KEYBINDING NAME="CreationClassName"><KEYVALUE
          VALUETYPE="string">EMC_IndicationFilter</KEYVALUE></KEYBINDING>
        <KEYBINDING NAME="Name"><KEYVALUE
          VALUETYPE="string">Filter1</KEYVALUE></KEYBINDING>
        </INSTANCENAME></INSTANCEPATH></VALUE.REFERENCE></PROPERTY.REFERENCE>
        <PROPERTY.REFERENCE NAME="Handler" PROPAGATED="false"
          REFERENCECLASS="CIM_ListenerDestination">
        <VALUE.REFERENCE>
        <LOCALINSTANCEPATH>
        <LOCALNAMESPACEPATH><NAMESPACE NAME="root"/><NAMESPACE
          NAME="interop"/></LOCALNAMESPACEPATH>
        <INSTANCENAME CLASSNAME="EMC_ListenerDestinationCIMXML">
        <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
          VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
        <KEYBINDING NAME="SystemName"><KEYVALUE
          VALUETYPE="string">thesystem</KEYVALUE></KEYBINDING>
        <KEYBINDING NAME="CreationClassName"><KEYVALUE
          VALUETYPE="string">EMC_ListenerDestinationCIMXML</KEYVALUE></KEYBIND
          ING>
        <KEYBINDING NAME="Name"><KEYVALUE
          VALUETYPE="string">test1</KEYVALUE></KEYBINDING>
        </INSTANCENAME></LOCALINSTANCEPATH></VALUE.REFERENCE></PROPERTY.REFERENCE>

        </INSTANCE>
        </IPARAMVALUE>
      </IMETHODCALL>
    </SIMPLEREQ>
  </MESSAGE>
</CIM>
```

Figure 37 **A CIM-XML request defining a new indication subscription (`CIM_IndicationSubscription`) using the filter and destination previously defined**

```
<?xml version="1.0" encoding="utf-8" ?>
<CIM CIMVERSION="2.0" DTDVERSION="2.0">
  <MESSAGE ID="2007:2:22:8:0:42:26:10" PROTOCOLVERSION="1.0">
    <SIMPLERSP>
      <IMETHODRESPONSE NAME="CreateInstance">
        <IRETURNVALUE>
          <INSTANCENAME CLASSNAME="EMC_IndicationSubscription">
            <KEYBINDING NAME="Filter"><VALUE.REFERENCE><INSTANCEPATH>
            <NAMESPACEPATH>
            <HOST>localhost</HOST>
            <LOCALNAMESPACEPATH>
            <NAMESPACE NAME="root"/>
            <NAMESPACE NAME="interop"/>
            </LOCALNAMESPACEPATH>
            </NAMESPACEPATH>
            <INSTANCENAME CLASSNAME="EMC_IndicationFilter">
            <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
              VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="SystemName"><KEYVALUE
              VALUETYPE="string">systemname</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="CreationClassName"><KEYVALUE
              VALUETYPE="string">EMC_IndicationFilter</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="Name"><KEYVALUE
              VALUETYPE="string">Filter1</KEYVALUE></KEYBINDING>
            </INSTANCENAME>
            </INSTANCEPATH>
            </VALUE.REFERENCE></KEYBINDING>
            <KEYBINDING NAME="Handler"><VALUE.REFERENCE><INSTANCEPATH>
            <NAMESPACEPATH>
            <HOST>localhost</HOST>
            <LOCALNAMESPACEPATH>
            <NAMESPACE NAME="root"/>
            <NAMESPACE NAME="interop"/>
            </LOCALNAMESPACEPATH>
            </NAMESPACEPATH>
            <INSTANCENAME CLASSNAME="EMC_ListenerDestinationCIMXML">
            <KEYBINDING NAME="SystemCreationClassName"><KEYVALUE
              VALUETYPE="string">CIM_System</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="SystemName"><KEYVALUE
              VALUETYPE="string">thesystem</KEYVALUE></KEYBINDING>
            <KEYBINDING NAME="CreationClassName"><KEYVALUE
              VALUETYPE="string">EMC_ListenerDestinationCIMXML</KEYVALUE></KEY
              BINDING>
            <KEYBINDING NAME="Name"><KEYVALUE
              VALUETYPE="string">test1</KEYVALUE></KEYBINDING>
            </INSTANCENAME>
            </INSTANCEPATH>
            </VALUE.REFERENCE></KEYBINDING>
          </INSTANCENAME>
  ...
```

Figure 38 **A partial CIM-XML response to the successful creation of an indication subscription**

4. Modify an instance of an indication subscription (`CIM_IndicationSubscription`) using the standard `ModifyInstance` CIM intrinsic operation.
5. 5. Delete an instance of an indication subscription (`CIM_IndicationSubscription`) using the standard `DeleteInstance` CIM intrinsic operation.

Secure indication delivery

ECOM can deliver indications over SSL. In order to receive indications from ECOM over SSL, a client must set the Destination property of CIM_ListenerDestinationCIMXML in the format of:

https://<ip_address>:<secure_port>

The indication recipient must listen on the secured port specified. If the indication listener requires certificate validation to enhance security, then ECOM's certificate MUST be signed and trusted by the listener's private key in order to receive the indications from ECOM over SSL.

Known limitations

The following are limitations in ECOM's support for CIM indications.

- ECOM does not support the modification of filters. To change a filter, you must first delete it and then create another.
- ECOM does not currently support predefined indication filters. Filters must be defined by client requests.