# Using Lanifex DMO to Model Information Assets, Risks and Policies

| | |
|---|---|
| Author : | Paul Gillingwater |
| Created: | 19 June 2005 |
| Updated: | 20. Jun. 2005 |
| Version: | 1.2 |
| Path: | J:\01_Customers\Lanifex\EventHorizon\Project Documents\LFX_RiskAssets.odt |

## Table of Contents

## Introduction

This document describes how the Lanifex Database of Managed Objects (DMO) may be used for modeling of Security Relationships within an enterprise. The DMO is a tool used to document objects within an information system or organization, so that implicit or explicit structure and relationships may be easily captured. In addition to objects, the DMO also allows a wide variety of other information to be documented, such as Information Assets, Policies and Procedures, Processes and Dependencies. Once the information is modeled, it can then be used in conjunction with other software to perform automated auditing and controlling, as well as risk analysis and statistical reporting.

Originally designed as a documentation tool, DMO is ideal for collecting together all documents related to an information system, supporting both planning and operational activities. During the past years, it has grown beyond its original purpose to include auto discovery of networks and policy compliance monitoring, as part of a comprehensive Information Security Management System. The DMO forms the heart of the Lanifex Event Horizon[1] security monitoring system, and is also built-in to the Lanifex Policy Compliance Module, which is used to manage compliance of organizations with the ISO17799 security standards.

1 Event Horizon is a separate product available from CSO Lanifex GmbH, which automates network and host security and vulnerability monitoring, storing information in the DMO.

## Data Structures within DMO

This part of the document describes the fundamental parts of the DMO, including definitions, instances and attributes.

The DMO system is implemented on top of a LAMP (Linux Apache Mysql PHP) system, and is based on the LFXlib programming library, developed by CSO Lanifex GmbH. This library includes the possibility of defining access controls over individual objects within the system, and supports a complex User/Group mechanism of rights administration, with the possibility of defining groups as members of other groups.

### Object Definitions

DMO is a general object-like[2] database which allows any type of object to be modeled within an information system or enterprise. An object within DMO is known as a Definition. All Definitions within DMO are arranged into a tree hierarchy with a single root, known as the "Definition Tree."

The table below shows the top-level objects which are in the current definition tree.

| | |
|---|---|
| **Application** | All software systems and applications are defined by this object. |
| **Business Services** | This definition is used to describe the Business Services which are supported by any combination of processes, hardware, network services and infrastructure objects defined within the DMO. This is the highest level of the dependency tree. |
| **COBIT Processes** | This hierarchy of definitions is used to document the COBIT processes within the enterprise. |
| **Information Asset** | This is some type of Asset used as part of a Risk Assessment. Assets will be associated with policies. |
| **Infrastructure Object** | This object includes all parts of the infrastructure which are not computer-related. |
| **ITIL Process** | ITIL is the Information Technology Infrastructure Library. Two of its 40 volumes relate to Service Support and Service Delivery. |
| **Location** | The physical location of some object. This is usually a building of some type. |
| **Network Object** | All objects within the network are defined underneath this object |
| **Policy Object** | This definition and its children define the various Security Policies which are implemented in an organization. |
| **Processes** | All network and system-related processes that must be managed are defined here. |
| **Unassigned** | Objects which are unknown or otherwise unassigned will be placed here. |

The definitions provided within the DMO are extensible by the administrator, with no restriction on their number. The only requirement is that the definition must be placed somewhere in the existing tree, and that every new object added has only a single defined parent definition.

Definitions are identified by their name (which must be unique) and their place in the Definition Tree. The name components in a definition path are separated by "//" characters. The root of the tree is the name of the information system or organization being modeled by the DMO. For example, "CSO Lanifex GmbH//Network Object//Node//Firewall." Child definitions should be

2 At the lowest level, DMO uses SQL within a relational database, but implements an object-relational model on top of this to support more sophisticated relationships.

related to their parents in some logical manner, implying a super-ordinate relationship.

The choice of definitions is important when designing the Definition Tree. One principle is that a definition should migrate to the highest level that is appropriate for the object being modeled. For example, a “Firewall” is a child of “Node”, which means some type of network-connected device. Typically, every “Node” has one (and only one) IP address. However, we know from experience that most firewalls have multiple IP addresses, therefore this might violate consistency of the model. To solve this, we define a new definition called “Firewall Network Interface”, which can then be made a child of the “Firewall” object. This in turn violates a principle of least duplication, because there are many types of devices with network interfaces, with no real way to distinguish between them. Therefore, we create a definition called “Network Interface”, which is a direct child of the “Node” object – and the firewall can then be assigned[3] as many instances of the “Network Interface” object as are required to model all its interfaces.

3 Linking an instance of a firewall to several instances of network interfaces is achieved by making the firewall instance the parent of each network interface instance. Note this is the parent of the INSTANCE, not the parent of the definition.

## Definition Attributes

Every definition within the DMO also has Attributes. There are three types of attributes: "Definition Core Attributes", which are available by default by every definition; "Instance Core Attributes", which are attached to every instance; and regular "Instance Attributes", which are assigned by the administrator to specific definitions, and are given values when linked to instances of those definitions.

The core attributes shared by all definitions are listed in the following table:

| **Name** | **Purpose** |
|---|---|
| Name | The unique name of this definition. |
| Revision | Used for change control within the definition. Typically this will be incremented every time the definition is edited. |
| Number | Every definition has a unique number, based on ASN.1. This allows the definitions to be used in an OID as part of SNMP. The number describes the hierarchy, e.g., 1.2.7.5. |
| Description | Text that describes what type of object is defined. |
| Owner | This is a user within the LFXlib system, that specifies who is the owner of the object. The owner has all rights over the definition, and whenever a new instance is created from the definition, it inherits the owner from the definition. |
| Group | This is a group within the LFXlib system, that defines various rights over the object. When an instance is created it will inherit the group from the definition. The rights for users are defined in terms of their membership of groups. Note that groups may also be members of groups, allowing for "function" groups and roles as well. |
| Type | This text may be used to classify or group the definition. Typically, the types may be taken from the root-level definitions, but other information may also be used. |
| Hidden | This attribute may be set as a boolean (1=true, 0=false) to hide the definition from the display of the definition tree. Only administrators will be able to see hidden definitions – all other users will not see them in the tree, allowing the tree to be pruned. Judicious use of this attribute will simplify DMO for other users. |
| Parent | Every definition in the tree must have a single parent defined, which is another definition. The only exception is the root definition, which is typically defined as the overall system or organisation being modeled. This field is required. |
| Definition Reporting | Checkbox which when enabled, will tag this definition for inclusion in the DMO System Reports, which will allow administrators to monitor such definitions more closely. |
| Show in Summary | If this checkbox is checked, this definition will be easily accessible on the system summary page. |

## Object Instances

Definitions only become useful when they are made specific. This is achieved by creating an "Object Instance" (also known as "Definition Instance.") As an example, "CSO Lanifex System//Network Object//Node//Host//Laptop" is a definition, which will have several instances, such as "Acer Travelmate 620", each with its own unique serial number and inventory number. Instances may be moved from one definition to another definition – but if this is done, care should be taken to ensure that the attributes are matched, so that no information is lost.

The core attributes shared by all instances are shown in the table below:

| **Name** | **Purpose** |
|---|---|
| Name | The unique name of this instance. |
| Date | The date that this instance was created within the DMO. |
| Revision | Whenever an instance is edited or changed, the revision number will automatically be incremented. |
| Serial | Instances may be assigned a serial number. |
| Location | The location, either logical or physical, where the instance may be found. |
| Inventory | An inventory number. This may be imported from an existing inventory system. |
| Purpose | The original purpose or use of this instance. Typically, this will not change over time, but will remain static. |
| Notes | Many system components will go through a life cycle, which includes pre-installation, production and decommissioning. This field is often used to record the state of the instance, or any special information . |
| Type | A general field which may be used to organize instances. An existing type may be selected, or a new one added when the instance is created. |
| Owner | This is a user within the LFXlib system, that specifies who is the owner of the instance. The owner has all rights over the instance. Normally the owner is inherited from the definition when the instance is created. |
| Group | This is a group within the LFXlib system, that defines various rights over the instance. When an instance is created it will inherit the group from the definition. The rights for users are defined in terms of their membership of groups. Note that groups may also be members of groups, allowing for "function" groups and roles as well. |
| Parent | This is the parent instance of the current instance. This field is optional, and is different from the Definition Parent, which must follow a strict hierarchy. By choosing the right parent for an instance, instance trees may be created that can model a hierarchy or system of grouping. |

It is important to distinguish between the single Definition Tree (which all definitions must appear in), and the existence of multiple instance trees. The instance trees may be built dynamically, and do not have to follow the same parent relationships which are shared by their definitions. The only restriction is that an instance may only have a single instance as its parent, i.e., it is a true inverted tree. Furthermore, instances may not exist in more than one instance tree at the same time.

Attributes are inherited from the parent definition; there is no inheritance of information via the parent instance, except in some specific cases, such as owner/group information on discovered instances.

## Instance Attributes

In addition to the core instance attributes, an instance may be assigned values of properties associated with its definition. These properties are known as "Instance Attributes", and may be inherited from parent definitions. Normally, the instance attributes for an instance are empty or non-existent. The range of available attributes for an instance is taken from the definition.

The attributes typically consists of a text field, which is free formatted. In addition to this text, every instance attribute may also have a file attached, allowing for any information to be stored. Note that only one file may be attached to a specific attribute of a specific instance; if multiple files are required, they must each be attached to their own attribute.

Furthermore, attributes cannot be duplicated – an instance will have only one "IP Address" attribute for example. If the model seems to require duplicate attributes, then it means that the model is incomplete, and instead one or more child definitions should be created – each with its own attribute. Due to inheritance, the same attribute is available for its child definitions as well. The example is provided of a UNIX server. If this server has a single IP address, then we can attach the "IP Address" attribute directly to the server instance – but if it is a multi-homed host, or has a logical interface with an additional address, then we need to create instances of the "Network Interface" definition, and assign each instance the server as the instance's parent.

Two instance attributes are provided by default to all definitions, which are "Support Documentation" and "Instance Picture." Both of these attributes are designed to have attachments.

The power of DMO is that the administrator may define an unlimited number of attributes, and attach them to any definition as required. This makes the system very flexible for modeling the information architecture.

## Specialized Attributes

In addition to the attributes described above, instances have several other very specialized attributes which are used to model relationships and store additional information.

These attributes are described in the following table:

| **Attributes** | **Purpose** |
|---|---|
| URLs | This attributes allows a URL to be added to an instance. Any number of URLs may be added to any single instance. In addition to the URL, each has a text description, owner and group. An important feature of this attribute is that URLs are stored in a single repository, which makes it easy for them to be reused. This saves duplication, and means that URLs may be linked to any number of instances.<br><br>Because the information in this attribute is a URL, it will be shown in the Web interface as a hyperlink, making it very easy to link to existing documents, Web interfaces or other information systems. |
| Dependencies | A dependency relationship may be defined between any two instances. The direction of the dependency may be "x depends on y" or "y depends on x". Dependencies may be chained, i.e., they can link several instances together. An unlimited number of dependencies may be added to a single instance, in both directions. In the current DMO version, it is possible to add information to a dependency (e.g., its type, or some documentation) by creating a new definition called "Dependency", and making the dependency relationships pass through this new object. (A later version will have more flexible dependency relationships.) |
| Intrinsic Link | This is a special type of link that functions as a URL, but with the DMO instance id of the current instance being passed as a parameter in the URI. This allows connection from the instance to an external system. Once an intrinsic link is defined, it is available for ALL instances. |
| Children | These are not really attributes; instead, they are instances which have the current instance as their parent instance. |
| Events | This is related to the Event Horizon system. Any security-related event that may be linked to the current instance will be displayed. (Only the most recent 10 events are shown.) |

## Modeling Information Assets, Policies and Risks

The starting point for any effort to model risks, threats and exposures is first to prepare a list of the Information Assets, which are the parts of an information system which contain value, and which should be protected.

When conducting a formal Risk Assessment methodology, such as OCTAVE[4], there are typically three phases:

1. Build Asset-based Threat Profiles
2. Identify Infrastructure Vulnerabilities
3. Develop Security Strategy and Plans

Any organization following a risk assessment will find it critical to categorize and prioritize its risks, which can only be done when there is a knowledge of which information systems and assets are the most valuable. This value can be expressed in terms of cost of replacement (in case of loss or corruption of data), cost of exposure (e.g., losses caused by inadvertent disclosure) and potential revenues (such as material being prepared for sale.) Critical information assets are not always obvious, as some information may be required for operations, that when missing or damaged will greatly degrade services, leading to indirect costs.

Within DMO it is possible to follow any one of several different approaches to this modeling process. Some customers may prefer to think in terms of physical machines, assuming that the data and services stored on those systems are congruent in risk. Other customers might wish to model in terms of Business Activities, and then look at the underlying infrastructure and data required to support those activities. Some customers may also wish to take a more geographical or organizational approach to their modeling, perhaps assuming that their services are duplicated over many branches, each with similar risk and protection profiles.

Our recommendation is to make the Information Asset model as simple as possible (but not simpler.) When the time comes to model dependencies and other relationships, the model will soon be complex enough. Each asset will of course have its own class of risks, threats and vulnerabilities associated with it, and a group of policies which should be applied to reduce or control those risks.

The minimum set of definitions we recommend are four:

1. **Asset**
   This is a business-based set of information and its supporting information systems (IS) which have clearly defined persons responsible for its maintenance, integrity and security.
2. **Asset Group**
   This definition is a way of grouping similar information Assets together. The instances of Asset will each have an Asset Group as their parent.
3. **Policy**
   This is a set of operational controls, which may be monitored for compliance either manually or automatically.
4. **Policy Group**
   This definition is used to group together related sets of Policy.

With the above definitions, we propose that links will be made between Asset Groups and Policy Groups. These links will be expressed as operational procedures or security controls, which can be used to verify the group of policies is being correctly applied to the group of assets.

4 Operationally Critical Threat, Asset, and Vulnerability Evalation is a risk assessment methodology developed for conducting self-directed security evaluations. It was developed by CERT.

In the above model, it should be noted that an Asset cannot be a member of more than one Asset Group at the same time, and that a Policy cannot be a member of more than one Policy Group at the same time.

This simplified model imposes these restrictions because it uses the Instance Tree technique to model the relationships, and group membership is modeled by the parent instance relationship. If a more sophisticated model is required, then instead of using the instance parent, an alternative approach is to use a new Instance Attributes, which will contain the name of the Policy Group (or Asset Group). This still restricts each attribute to having only one value – but there is an unlimited number of attributes available, which means that new ones may be created for each group.

Next, the relationship between the Asset Group and the Policy Group should be modeled. Using an object-relational notation, we might say that "a specific Asset Group HAS a specific set of policies, linked into one Policy Group." Therefore, it would be intuitive to make the Asset Group a parent of the Policy Group – however, this would limit to one the number of times a Policy Group could be used, therefore we instead recommend using the Dependencies feature of the instance. Thus, we can say that a specific Asset Group "depends on" a specific Policy Group – and we can define as many dependencies as are required.

## Security Policies

Security Policies may be stored within DMO as instances of various types. Presently, this is designed to follow the ISO17799:2000 standard, but could be adapted to other standards, including ones unique to the organization. Each policy object may have documents attached to it (via its attributes) which further explain the policy, and attributes may also be used to track compliance.

Within the Lanifex Policy Compliance Module[5], we store policies following ISO17799:2000, and link this to a set of workflow procedures that perform audits based on a defined set of controls. These audits are managed as questionnaires, which are sent automatically to the person or group identified as responsible for the controls associated with the information asset.

Based on the results of the compliance audit (including escalation for non-participation in the questionnaire), reports are prepared on the level of compliance achieved, which provide a fast method for an organization to continuously monitor its level of information security process maturity[6].

5 This is a separate product, based on ISO17799:2000, which uses DMO to store information.
6 We recommend use of the Systems Security Engineering Capability Maturity Model (SSE-CMM), as a standard method of assessing how well an organization has control over its security-related processes.

## About Risk

According to our understanding, there is a total of five things which can be done about risk:

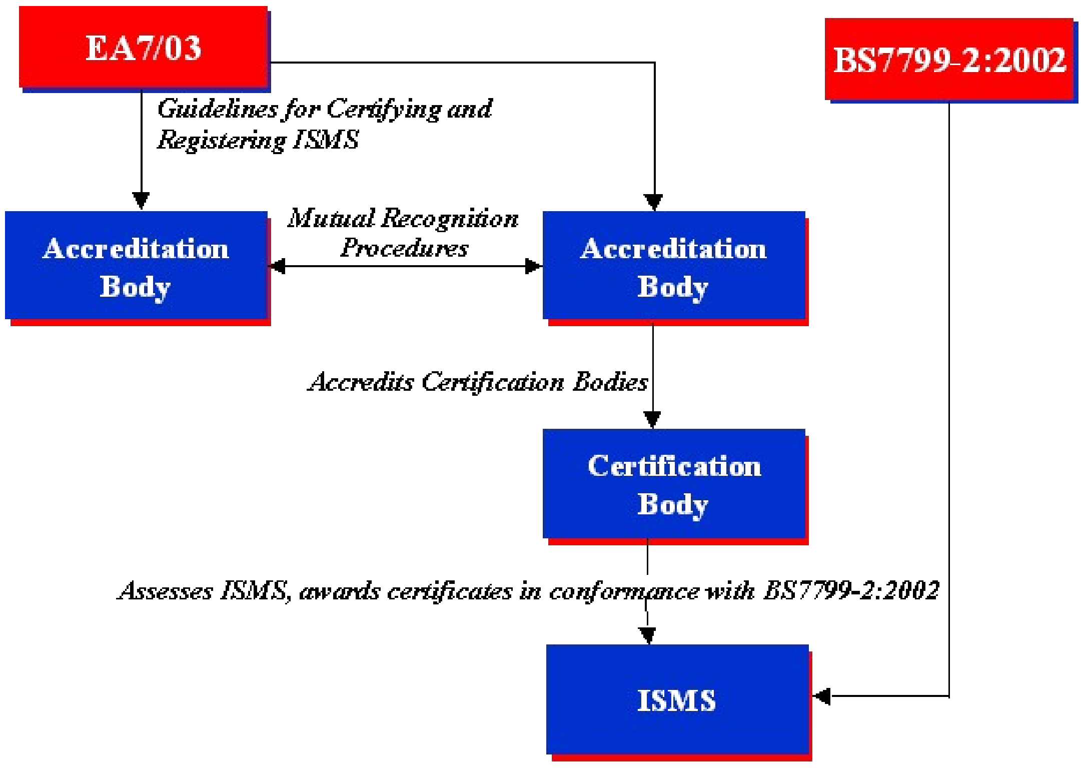
1. **Mitigate (Reduce) Risk**
   This means finding some operational controls which can reduce the exposure or expected results from a specific risk;
2. **Avoid Risk**
   If the risk is only incurred when a specific action occurs, then avoid performing the action. This may not make business sense—it all depends on the risk/reward ratio.
3. **Ignore Risk**
   Most businesses unfortunately do not give due consideration to the risk. Thus, it comes as a surprise to them when the risk is realized; some businesses do not survive;
4. **Transfer Risk**
   The classical insurance model is an ideal example of risk pooling and transfer;
5. **Accept Residual Risk**
   You've done everything to reduce the risk, but if you wish to stay in business, there is some residual (remaining) level of risk that is simply accepted as a cost of doing business.

Once an Asset has been identified, and stored in the DMO, it will then be linked to specific policies, which govern how that asset will be controlled in terms of the risks. One essential document which should be attached to each asset is a "Statement of Applicability" (SoA), which defines WHY specific security controls are being associated with the assets. In addition, the ISO17799 standard requires that any controls NOT selected for an asset should also have a SoA to explain their absence. Typically, controls are related back to the risk assessment, however controls may also be related back to the ISMS policy, as used in Common Criteria.

## Certification

The ISO 17799:2000 standard has no support for certification, as it was based on the older BS7799 Part 1. With the introduction of BS7799 Part 2, which governs Integrated Security Management Systems (ISMS), it became possible to perform audits against the British Standard. The newer ISO 17799:2005 now intoduces the possibility of performing audits in future, but at present (2005) the only realistic option for audit and certification of the security system is based on the European co-operation for Accreditation document EA7/03, which allows ISMS to be certified based on BS7799 Part 2.   This is likely to be extended in future also for ISO 17799:2005.

The diagram below shows the certification process.


EA7/03
BS7799-2:2002
Guidelines for Certifying and Registering ISMS
Accreditation Body
Mutual Recognition Procedures
Accreditation Body
Accredits Certification Bodies
Certification Body
Assesses ISMS, awards certificates in conformance with BS7799-2:2002
ISMS