

Oval 5.x Proposal Analysis

Contributors:
Ken Lassenen, Patchlink.com



1. Table of Contents

- 1. Table of Contents2
- 2. Objectives5
 - 2.1 Notation5
 - 2.2 Terms6
- 3. Major Oval Components7
 - 3.1 Example of Xml1
 - 3.2 <tests>1
 - 3.3 <objects>1
 - 3.4 <states>1
 - 3.5 <vars>1
 - 3.6 <definitions>1
 - 3.7 <solutions>1
 - 3.7.1 Multiplicity of <solution>s2
 - 3.7.2 Multiplicity of <alternative>s2
 - 3.7.3 Need for additional <criteria> / <tests>3
 - Extension to handle individualistic tests <policy_test>3
 - 3.7.4 Resolving the <alternative>'s4
 - 3.7.5 User Interaction4
 - 3.7.6 Preference5
 - 3.8 <remedies>5
 - 3.8.1 Security of <remedy>5
 - 3.8.2 Reboot in <remedy>5
 - 3.8.3 <remedy> Failure5
 - 3.8.4 Location of <remedy>6
 - 3.9 Proposed Structure for Raw <remedy>6
 - 3.10 Proposed Elements6
 - <solutions>6
 - Parents:6
 - Children:6
 - <solution>6
 - Parents:6
 - Children:6
 - <resolutions>6
 - Parents:6
 - Children:7
 - <resolution>7
 - Parents:7
 - Children:7
 - <alternatives>7
 - Parents:7
 - Children:7
 - <alternative>7
 - Parents:7
 - Children:7
 - <logic>8
 - Parents:8
 - <resolves>8

- Parents: 8
- Children: 8
- <userinteraction>..... 8
 - Parents: 8
 - Children: 8
- <prompt>..... 8
 - Parents: 8
 - Children: 8
- <permission>..... 9
 - Parents: 9
 - Children: 9
- <criteria> 9
 - Parents: 9
 - Children: 9
- <eulas> 9
 - Parents: 9
 - Children: 10
- <eula> 10
 - Parents: 10
 - Children: 10
- <lang> 10
 - Parents: 10
 - Children: 10
- <remedies> 10
 - Parents: 10
 - Children: 10
- <remedy> 11
 - Parents: 11
 - Children: 11
- <contents> 11
 - Parents: 11
 - Children: 11
- <package> 11
 - Raw Program flow 12
 - Parents: 13
 - Children: 13
- <verifications> 13
 - Parents: 13
 - Children: 13
- <verification> 14
 - Parents: 14
 - Children: 14
- <locations> 14
 - Parents: 15
 - Children: 15
- <uri> 15
 - Parents: 15
 - Children: 15
- <component> 15
 - Parents: 15

- Children:15
- <verification>15
 - Parents:16
 - Children:16
- <failure>16
 - Parents:16
 - Children:16
- <criteria>16
 - Parents:16
 - Children:16
- <file>16
 - Parents:17
 - Children:17
- <script>17
 - Parents:17
 - Children:17
- <script_object>17
 - Parents:18
 - Children:18
- <script_test>18
 - Parents:18
 - Children:18
- <policy_test>18
 - Parents:18
 - Children:18
- 3.11 Import Utilities18
- 4. Tests Status / Depreciations19
- 5. Product Identification20
 - 5.1 Adapting the CMSI Standard21
- 6. General Issues.....23
 - 6.1 Implementation Logic:23
- 7. Revision History24
- 8. Notes on Taxonomy of Terms.....25

2. Objectives

This document is a work in progress of notes on extending OVAL to cover solutions and remedies. It is hoped that this will lead to inclusion of **solutions** and **remedies** in the next release of Oval.

Some specific objects are:

1. Establishing taxonomy for nodes and attributes.
2. Verifying general structure and relationships
3. Identifying special cases that are not adequately addressed

Another future extension of OVAL may be in the direction of incident information, either adapting existing proposals (preferred) such as INCH (Incident Handling Working Group), or generating a proposal.

There is, and will likely continue to be overlap between various standards, for example, vulnerability and security issues. This could be described as:

- A vulnerability / definition are a component that in its normal state has risk. For example, a security login interface that has an embedded backdoor login/password, or which automatically installs a default one
- A security item is a setting that could result in a component having risk. For example, not having a password.
- When the default install state of a component has risk, but the component is sound if in the appropriate state – we have a fuzzy classification.

It is suggested that another addition would be to add support for various standards through an explicit reference, rather than implicit (i.e. use the same id as another system). For example:

- @cve_ref
- @msft_ref
- @cisco_ref
- @xccd_ref
- @csmi_ref

This would facilitate *increased flexibility* with vendor implementations, especially if a central cross reference repository is maintained and used to update all references with their equivalent in other systems.

This is intended as a discussion document, the views expressed are not necessarily the opinion of PatchLink.com, but that of the author. Please send comments to:

Ken.Lassesen@gmail.com, or
KenL@exMsft.com

A community board for this proposal is available at: <http://oval.reddwarfdogs.com/Discussion>

2.1 Notation

Notation	Description
----------	-------------

<name>	Indicates a concept that is stored as a node
@name	Indicates a concept that is stored as an attribute

2.2 Terms

- system(s) administrator** – indicates the entity that administers a PC or a set of PCs. The entity that chooses from the set of possible solutions to fix a <definition>. This may be a person or an agent (such as a software agent).

3. Major Oval Components

The following describes changes or enhancements to the existing Oval 5.0 Schema in order to support <solution> and <remedy>. This extends Oval from just identifying issues (<definition>) into the next logical problem space, correcting the issues.

An issue (<definition>) should have a <solutionⁱ> set consisting of several <alternative> ways to reach a <resolution>. Each <alternative> way is done through a <remedyⁱⁱ> which may:

- Execute other <remedy>s
- May be subject to <criteria> that identify which <alternative>s are *feasible*
- May have different @weightⁱⁱⁱ of importance for the system(s) administrator.

One aspect of <solution> and <remedy> is that there may be system(s) administrator preferences that can modify the @weight of each <solution>. This means that there may be a need to create an import tool that retains changes when updates are persisted. The items that should be persisted are identified in the Import Tool section below.

For the purpose of taxonomy, <solution> and <remedy> are deemed to be *Installers*.

Figure 1 Difference between Installers and Package Management¹

Package Management System	Installer
Typically part of the operating system.	Each product comes bundled with its own installer.
Uses a single installation database.	Tracks its own installation.
Can verify and manage all packages on the system.	Only works with its bundled product.
Single package management system vendor.	Multiple installer vendors.
Single package format.	Multiple installation formats.

¹ From <http://en.wikipedia.org/wiki/Installer>

3.1 Example of Xml

The following is a sample of what the XML may look like to provide some concreteness to the descriptions below.

XML 1 Example of proposed New Elements

```
<?xml version="1.0" encoding="utf-8" ?>
<oval>

  <solutions>
    <solution id="oval:patchlink.com:sln:2321"
      comment="">
      <resolutions>
        <resolution definition_ref="oval:org.mitre.oval:def:9"/>
        <resolution definition_ref="oval:patchlink.com:def:912"/>
      </resolutions>
      <alternatives>
        <alternative remedy_ref="oval:patchlink.com:rmd:1720310">
          <criteria test_ref="oval:org.mitre.oval:tst:19"
            weight="0.2"></criteria>
          <resolves definition_ref="oval:patchlink.com:def:76732"/>
          <userinteraction operation="AND">
            <prompt eula_ref="oval:org.mitre.oval:ela:534"
              prompt="acceptdecline"/>
            <permission type="pick"/>
            <permission type="install"/>
            <permission type="reboot"/>
            <permission type="delay"/>
          </userinteraction>
        </alternative>
        <alternative remedy_ref="oval:org.mitre.oval:rmd:534">
          <criteria test_ref="oval:org.mitre.oval:tst:19"
            weight="0.3"></criteria>
          <criteria test_ref="oval:org.mitre.oval:tst:434"
            weight="0.14"></criteria>
          <criteria test_ref="oval:org.mitre.oval:tst:434"
            weight="0.14"></criteria>
          <userinteraction operation="AND">
            <prompt eula_ref="oval:org.mitre.oval:ela:535"
              prompt="acceptdecline"/>
            <permission type="pick"/>
            <permission type="remove"/>
          </userinteraction>
        </alternative>
      </alternatives>
    </solution>
  </solutions>
</oval>
```

```

        </userinteraction>
    </alternative>
</alternatives>
</solution>
</solutions>
<remedys>
  <remedy id="oval:patchlink.com:rmd:1720311"
    comment="example of a RAW Remedy">
    <verifications>
      <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
    </verifications>
    <contents>
      <package type="raw">
        <locations @type="file">
          <location domain="Redmond.Microsoft.com"
            login="kenl"
            password="drGui92"
          >
            https://partners.corpnet.microsoft.com/vista/
          </location>
          <uri>
            http://library/vista/
          </location>
        </locations>
        <files>
          <file relpath="system32">
            <name>win32divx.dll</name>
            <verifications>
              <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
            </verifications>
          </file>
          <file relpath="system64">
            <name>win64divx.dll</name>
            <verifications>
              <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
            </verifications>
          </file>
          <file relpath="system16">
            <name>win16divx.dll</name>
            <verifications>
              <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
            </verifications>
          </file>
        </files>
      </package>
    </contents>
  </remedy>
</remedys>

```

```

<script type="cmd">
  <![CDATA[
XCOPY *.* %windir% /s /y /f]
]]>
  <verifications>
    <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
  </verifications>
</script>
</package>
</contents>
</remedy>
<remedy id="oval:patchlink.com:rmd:1720310"
  comment="example of a Remedy"
  hasreboot="true">
  <verifications>
    <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
  </verifications>
  <contentss>
    <package type="msi">
      <locations @type="file">
        <location domain="Redmond.Microsoft.com"
          login="kenl"
          password="drGui92"
        >
          https://partners.corpnet.microsoft.com/vista/
        </location>
      </locations>
      <uri>
        http://library/vista/example.msi
      </location>
    </locations>
  </files>
  <file
    type="source">
    <name>example.msi</name>
    <verifications>
      <verification type="md5">f15a39318cf10e8de5832da103e7fd80</verification>
    </verifications>
  </file>
</files>
  <verification when="after reboot">
    <failure remedy_ref="oval:patchlink.com:rmd:14846733"
      retries="10"
      wait="3600"/>
  </criteria operator="AND">

```

```

        <criteria comment="Software section"
            operator="AND">
            <criterion comment="Exchange Server 2003 (gold edition) is installed"
                negate="false"
                test_ref="oval:org.mitre.oval:tst:1367"/>
        </criteria>
        <criteria comment="Configuration section"
            operator="AND">
            <criterion comment="The exadmin HTTP virtual directory only allows Integrated Windows
Authentication"
                negate="false"
                test_ref="oval:org.mitre.oval:tst:1366"/>
        </criteria>
    </criteria>
</verification>
<failure wait="3600"
    comment="wait for a hour and keep trying for one day"
    retry="24">
    <component remedy_ref="oval:patchlink.com:rmd:148467344" />
</failure>
</package>
</contentss>
</remedy>
<remedy id="oval:patchlink.com:rmd:1720310"
    comment="example of a compound Remedy"
    failure="stop">
    <contents>
        <component remedy_ref="oval:patchlink.com:rmd:14846733" />
        <component remedy_ref="oval:patchlink.com:rmd:14941233" />
    </contents>
</remedy>
</remedys>
<eulas>
    <eula id="oval:org.mitre.oval:ela:534">
        <lang xml:lang="fr-fr">
            AVENANT AU(X) CONTRAT (S) DE LICENCE UTILISATEUR FINAL
            VISUAL STUDIO 6.0 PROFESSIONAL EDITION ET VISUAL STUDIO .NET ACADEMIC EDITION
            (Pour le Programme Microsoft Developer Network Academic Alliance- Adhésion réservée aux
            établissements d'enseignement secondaire)
            -----
            Le présent avenant (l'"Avenant") au(x) Contrat(s) de Licence Utilisateur Final Microsoft pour Visual
            Studio 6.0 Professional Edition (le "CLUF VS 6.0") et Microsoft Visual Studio.NET Academic Edition (le "CLUF

```

VS.NET") (le(s) "CLUF(s) VS") constitue un contrat entre Microsoft Corporation ("Microsoft") et un Utilisateur Éducation Autorisé que Microsoft a autorisé à participer au Programme Microsoft Developer Network Academic Alliance - Adhésion réservée aux établissements d'enseignement secondaire (le "Programme").

</lang>

<lang xml:lang="en-ca">

Amendment to Master End-User License Agreement for
Microsoft Software, The Microsoft Developer Network Subscription
(For the Microsoft Developer Network Academic Alliance Program)

This amendment (the "Amendment") to the Master End-User License Agreement for Microsoft Software, The Microsoft Developer Network Subscription (the "EULA") is a legal agreement between Microsoft Corporation ("Microsoft") and a Qualified Educational User (as hereinafter defined) approved by Microsoft for participation in the Microsoft Developer Network Academic Alliance Program ("MSDN Academic Alliance Program").

</lang>

</eula>

</eulas>

</oval>

3.2 <tests>

A <test> is a defined Oval Component and retrieves information from the client environment. The information is typically the state of an object, or a logical evaluation of the states of multiple objects.

A new *usage* is added to <tests> by using it in an <alternative>'s <criteria> to determine if the <alternative>'s <remedy> is *feasible*. This determines the details needed to select the appropriate remedy, for example, what language should the UI components be in? What is the base product or operating system being modified? (a <definition> may apply to multiple versions of a product, so the test may apply to 5 versions of Windows and 4 major releases in each of 10 languages. The solution may need to be both specific to the Windows version, the release version and the language (thus a 1:200 ratio between <definition> and <solution> could occur).

Furthermore, two additional type of tests are suggested which could be called <policy_test> and <script_test>. This obtains information such as licensing permissions or upgrade permission in determining the feasible subset of <alternative>s.

For further information see Oval Documentation.

3.3 <objects>

This is a component of a test and defines some aspect of the client environment.

For further information see Oval Documentation.

3.4 <states>

This is a component of a test and defines a property of some aspect of the client environment.

For further information see Oval Documentation.

3.5 <vars>

This is a component of a object or state that is used for normalization of data storage.

For further information see Oval Documentation.

3.6 <definitions>

A definition is a defined Oval Component that describes an issue. It uses members of oval:<tests> to determine if this issue exists.

For further information see Oval Documentation.

3.7 <solutions>

A solution is a proposed Oval Component that modifies an *issue* (<definition>). For a particular issue, there may be many ways of modifying it; these ways are called <alternative>^{iv}s. An

<alternative> is a container for the changes to the client environment. These changes are called a <remedy>.

An <alternative> should always have a <criteria> because there may be many possible choices (removal, update, upgrade, disabled).

3.7.1 Multiplicity of <solution>s

One <solution> may remedy many <definition>s. For example a version of Office may use several software-components that each have their own <definition>, for example, database drivers. Each of these database drivers may have its own <solution>s that addresses only one specific driver. A <solution> to Office may be also correct these items, or may not – for example,

- <alternative> 1 – upgrades to the latest version (which also upgrades all of the database drivers – everything is fixed)
- <alternative> 2 – disables the item described in the <definition>. None of the database drivers are changed, and thus *not* a solution to this.

In order to capture *side-effects* of an <alternative>, <alternative>s may have <resolves> that identify other definitions that this <alternative> may fix.

To illustrate this, consider that the results of tests, found there are issues with def:1, def:2, def:3. The first step would be to identify the <solution>s that resolves them, that could be done by the following xpath:

```
//resolution[@ref_definition='def:1' or @ref_definition='def:2' or
@ref_definition='def:3']::parent::parent
```

This would return all of the <solution>s that solve one or more of these <definitions>. In that collection there may be solutions that will resolve all of them at one time. The system(s) administrator would need to pick the most appropriate one(s) according to guidance obtained elsewhere. The @weight is intended as a *primitive* default guidance mechanism which can be manually set, with *value-added* vendors providing more advance guidance components within <logic> that is vendor specific.

3.7.2 Multiplicity of <alternative>s

There may be multiple solutions for a definition. Some prospects of a solution are:

- Removal of an element in the environment
 - For example uninstalling a component
 - Removing of a file
 - Calling a virus check to delete some bits in a file
- Modification of an element in the environment
 - Changing settings or parameters
 - Replacing a component (upgrading version)
- Addition of an element in the environment
 - Adding a component (for example, a firewall)
 - Adding a setting or parameter (for example, turning on a fire wall)
- Multiple generation patches (aka superseded patches)
 - Later patches may contain upgrades that are not desired to be rolled out, hence the generations of patches should be available.
 - By assigning appropriate @weight the corporate administrator can cause the desired patch to be selected.

3.7.3 Need for additional <criteria> / <tests>

A solution may have additional <criteria> that must be true in order for this solution to be applicable. Some prospects for these tests (with possible consequences) are:

- Multiple versions of Office are installed, older versions have vulnerability.
 - → Remove older version of Office
- A DLL is installed in an application directory for some 3rd party application.
 - → If known application, upgrade application
 - → If not known, remove DLL from folder (may break application)
- If a patch has a prerequisite, then there may be two solutions:
 - one with a test that the prerequisite is there and
 - The remedy's components do not install the prerequisite(s)
 - the other, checking that the prerequisite is not there.
 - The remedy's components do install the prerequisite(s)
- If solution updates a UI component, then the required language
- Is the user licensed for this? (may require <policy_test>)

Extension to handle individualistic tests <policy_test >

In terms of system(s) administrator, some information may come from corporate licensing databases and how this information is obtained may be *individualistic* to the situation. It is unlikely, and may be undesirable, to have a global solution here, although one is described below:

One solution to this may be a test that causes an *intranet* web service (<http://oval.corporate/>) which passes in the *parent* identifier that holds the test (i.e. <definition>, <alternative>, <remedy>). The web service returns a true(0) or false(-1). In this case the test definition would appear to be similar to what is shown below and it may be liberally dropped into any <alternative> that *could* require corporate information.

```
<policy_test id="oval:org.mitre.oval:tst:1"
  version="1"
  check="at least one"
  comment="Generic Policy State">
  <object object_ref="oval:org.mitre.oval:obj:1"/>
</policy_test>

. . .
<policy_object id="oval:org.mitre.oval:obj:1000" version="1" >
  <uri servicetype="post"
operation="equals">http://oval.reddwarfdogs.com/service.aspx</uri>
</policy_object>
<policy_object id="oval:org.mitre.oval:obj:1001"
  version="1" operation="OR">
  <uri servicetype="webservice" failure="false" comment="stock call"
operation="equals">http://oval.service.asmx</uri>
  <uri servicetype="webservice" failure="false"
operation="equals">http://oval.reddwarfdogs.com/service.asmx</uri>
  <uri servicetype="webservice" failure="true"
operation="equals">http://oval.lasseesen.com/service.asmx</uri>
</policy_object>
```

If the call fails, then there may be alternative <uri> available to try as shown in the example above.

This allows a single test to handle all corporate queries in a low obtrusive way. System(s) administrator may:

- create this web service
- change the location in the test definition before distribution
- replace the test with a no-op test.

3.7.4 Resolving the <alternative>'s

As discussed above, there may be many <alternative>s in a solution.

An <alternative> may have a cost, -that is, something that *gets consumed*. This can have many dimensions, for example:

- Consumption of disk space for remedy files etc
- Consumption of available license(s)
 - For example, upgrading from Office 2000 to Office 2003
 - Is the user licensed for this already?
- Consumption of money/budget
 - Cost of a license or an upgrade

Some of these aspects will likely be corporation specific. The <policy_test> described above may be used to include or exclude an <alternative>, for example checking a database with Microsoft Volume Licensing information and seeing if there is available site licensing.

It is desirable to have rules for resolving the alternative, and to this end, <alternatives> may have a node type called <logic> which can contain *customized logic to resolve the issue*. This is a place holder for vendors to value-add to the Oval definitions for their products.

In lieu of custom logic, each <alternative> has a @weight calculated by adding up the @weight in each <criteria> of the <alternative>. The system(s) administrator can update the @weight to match IT/corporate policies and preferences. *Any oval administration tool should retain assigned values to the @weights when a new version of an existing object is distributed.*

The @weights assigned by the <solution> creator should follow the priorities listed below:

- Upgrade to the newest version is the first choice.
- Patching the existing version is the second choice
- Restricting the existing version is the third choice
- Disabling the existing version is the fourth choice
- Removing the existing version is the last choice.
- The **maximum weight** that can occur in a *distributed* <alternative> is 1.0. Weights over 1.0 indicate that it is a custom assignment by some system(s) administrator.
-

Remember – these values will likely be modified by the system(s) administrator.

3.7.5 User Interaction

Before a specific solution may occur there may be user interaction requirements that must be fulfilled. Some prospects are:

- Acceptance of one or more EULA
- Permission to reboot

- Permission to remove or install
- Permission to proceed (i.e. install now or delay for X hrs)
- Choice of solution (the user may pick a solution from several available solutions)
 - This may include time of install options {"at 8pm"}{"at midnight"}

3.7.6 Preference

One solution may be preferred over another solution for a specific corporation depending on the whims of their IT department. This is done by the modification of the @weight in the <alternative>'s <criteria>.

3.8 <remedies>

A Remedy is a proposed Oval Component that causes a change in the client environment. These changes may be done by downloading components and executing a script or thru some form of installation package.

Some examples of installation packages are:

- MSI on Windows
- DMG on Mac
- RPM on Unix

3.8.1 Security of <remedy>

A <remedy> provides a major opportunity for malware, Trojan horses or virus distribution. In order to prevent this, digital verifications are used on the individual files/script, and on the *remedy* itself. There is an intrinsic ability to verify with the *originator* of a remedy that it has not been tampered with and also establish accountability for the remedy.

3.8.2 Reboot in <remedy>

Some <remedy>s may get part of an installation done and then require a reboot to complete it. In some cases, there may be a need to do multiple reboots (this has been seen with some Microsoft issued MSIs). When there is a reboot, if there is additional processing then it is assumed that the remedy will automatically start the continuation of a <package>.

In the case of a raw.package, the raw.package should be divided into parts for each reboot with an appropriate verification to determine which part of the *compound* remedy to do next. The client is expected to retain which <alternative><remedy> is in progress. The <remedy>'s would have @hasreboot="true" (may be buried in the case of a nested remedy).

3.8.3 <remedy> Failure

A <remedy> may fail for many reasons (including insufficient permission). In *an ideal world*, the <test> should exclude all possible situations (except the user aborting the remedy), in reality, failures happen. Trying for the 1000th time to install the <remedy> is not a good solution. The <failure> node determines what should happen when a failure occurs.

The remedy in the failure node will typically be an information gathering and transmission remedy which could include reporting to a web service or sending an email. The remedy occurs after the specified retries are exhausted.

3.8.4 Location of <remedy>

The location of the remedy may be public or private. A remedy that requires or is subject to licensing will typically be at a private location. If the solution is issued with a site that needs authorization, but the authorization parameters are licensee specific, then “%user%”, “%domain%”, “%password%”. See <uri> below.

3.9 Proposed Structure for Raw <remedy>

The schema for a raw package needs to have all of the capabilities that exist in a MSI, dmg, or rpm package. In reviewing the contents of the packages, the simplest solution is to just support two components <file> and <script>.

In terms of the 3rd party products, they may include the ability to take a raw <package> and automatically generate the different package formats from the raw package.

3.10 Proposed Elements

<solutions>

The container for <solution>s.

Parents:

- <oval>

Children:

- <solution> (1-n)

<solution>

- @id [required] – string, in typical oval format with “sln”
- @comment [optional] – string plain English description

Standard oval child nodes: <notes>

Parents:

- <solutions>

Children:

- <resolutions> (1)
- <alternatives>(1)

<resolutions>

Container for <resolution>s.

Parents:

- <solution>

Children:

- <resolution> (1-n)

<resolution>

Identifies what a solution resolves.

- @definition_ref [required], string – a reference to an existing definition that this impacts
- @comment [optional], string – an explanation if needed

Parents:

- <resolutions>

Children:

- none

<alternatives>

A container for <alternative>s.

Parents:

- <solution>

Children:

- <alternative> (1-n)
- Any – to allow open-ended selection logic to be added according to system(s) administrator wishes.

<alternative>

Defines an explicit solution

- @id [required] – string, in typical oval format with “alt”
- @remedy_ref [required], string – a reference to an existing remedy that is to be used. This may be *top* level of a remedy tree.
- @comment [optional], string an explanation if needed

Parents:

- <alternatives>

Children:

- <userinteraction> (0-1) – 0, if silent install.
- <resolves> 0-n, other <definition>s that would also be corrected.
- <criteria> 0-1 – see <definition> for pattern
- <logic> 0-1

<logic>

Contents of this is any element. The content is determined by *value-added* vendors. Since the number of <alternatives> may be constantly changing (increasing),

Parents:

- <alternatives>

<resolves>

Identifies other <definition>s that will also be resolved if this <alternative> is implemented.

Attributes include:

- @definition_ref [required] string – a definition that is corrected by this <alternative>.
- @comment [optional] string – provides additional information if needed.

Parents:

- <alternative>

Children:

- None

<userinteraction>

Container for a collection of user interactions which may be required due to license requirements.

- @operation [optional] – default is AND. Typical Oval enumeration of @operation

Parents:

- <alternative>

Children:

- <prompt> (0-n) – handles the presentations of EULA exclusively
- <permission> (0-n) -- handles the permission to restart machine, etc
-

<prompt>

- @eula_ref [required], string, a reference to an <eula> that must be displayed to the user
- @prompt [optional] – one of the following
 - “acceptdecline” – default

Parents:

- <userinteraction>

Children:

- none

<permission>

Installer must seek permission from the user to do actions such as reboot, install, remove, delay. All of these items will result in a screen prompt that *should be localized* into the default language of the operating system by the installer.

- Type [required], one of:
 - “reboot” – computer may be rebooted
 - “install” – user must consent to the install
 - “remove” – user must consent to the removal
 - “delay” -- allows the user to delay the action for a period of time
 - “pick” – this may be presented to the use as a choice for solution (typically this will be added by the system(s) administrator).

Parents:

- <userinteraction>

Children:

- none

<criteria>

This is identical to the definition.criteria with the addition of a @weight property. They may be nested with the following logic applied to the @operation:

- “AND” – means sum the @weight
- “OR” – means take the maximum of the @weight

The criteria in this case may include tests for:

- Specific language being used on the OS
- Specific patch level on the OS

It is suggested that a set of standardized tests based on CMSI (Common Model of System Information²) be developed to prevent duplication of tests.

Parents:

- <definition>
- <alternative>
- <failure>

Children:

- <criterion> (0-n)
- <criteria> (0-n)

<eulas>

This is a container for <eula>s.

Parents:

² See http://www.cert-verbund.de/cmsi/data/cert_verbund_cmsi.xml#wxp

- <oval>

Children:

- <eula> (1-n)

<eula>

A eula is a piece of text that needs to be presented to the user. There may be different eula for different languages. Each <lang> contains one version.

- @id [Required]– string, in typical oval format with “ela”
- @comment [optional] – string

Parents:

- <eulas>

Children:

- <lang> (1-n)

May have oval:norte as children

<lang>

- @xml:lang [required]– the classic XML method of identifying the language. Enumeration

It is up to each client/implementation to resolve the fall back if the ideal language is not available.

Parents:

- <eula>

Children:

- CDATA or text (1)

<remedies>^v

The container for <remedy>s.

Parents:

- <oval>

Children:

- <remedy> (1-n)

<remedy>

A remedy causes a change in the user environment and should be *atomic*. To assist in insuring that it is atomic, a @style is required. Thus a solution to uninstall Word and install Word Perfect cannot be in a single remedy, there will be an <alternative> that contains a <remedy style="removal"> that uninstalls Word, and a <remedy style="full install"> that installs Word Perfect.

Attributes may include:

- @id [required] string - in typical oval format with "rmd"
- @comment [optional], string an explanation if needed
- @hasreboot [optional], Boolean – indicates that a reboot is required to complete this remedy.
- @failure [optional] – enumeration when multiple components are being installed
 - "ignore" – proceed to the next component
 - "stop" – do not install any more components.
- @style [required], one of:
 - "removal"
 - "upgrade"
 - "configuration"
 - "full install"
 - "patch"

Allowed child nodes includes oval <notes>.

Parents:

- <remedies>

Children:

- <verifications> (0-1) – provides verifications for the contents
- <contents> (1) – only one direct package is permitted in a remedy (atomic)

<contents>

This is the container for the remedy parts. The <remedy>'s <verifications> apply to the <contents>.

Parents:

- <remedy>

Children:

- <component> (0-n) – this happens when there are multiple remedies in one remedy.
- <package> (0-n) – only one direct package is permitted in a remedy (atomic)

<package>

A package is typically a *single file* that is self-contained for a change that is **used by a component** that exists on the computer. The oval-interpretter/client-agent is assumed to have the intelligence to locate or install the component that installs the package. No further

processing is required other than finding the executable name, giving it the file name (and optionally depending on the client-agent, additional parameters for items like silent install³).

A *raw* package is the exception because it consists of multiple files and an install <script>. A script is code that is passed to the operating system and directly executed. A raw package may be close to being a package, for example, it may contain a single file (.zip, tar, etc) and a script that unzips this file and then executes. It is generally recommended not to do individual files, but to use a file storage structure like zip or tar.

InstallAnywhere⁴ supports most systems but requires Java to be installed.

Attributes include:

- @type [required], enumeration one of:
 - “dmg”⁵ – Apple’s Disk Image, *application/octet-stream*
 - Includes: bzip2 (.dmg.bz2) and zip (dmg.zip)
 - “sdux”⁶ – HP-UX software distributor
 - “img” – Apple’s Disk Image
 - “jar” – Java based installation package.
 - “msi” – Microsoft Installer package
 - “solaris-patch” – Solaris Patch
 - “pkg”⁷ – Solaris Package
 - “rpm”⁸ – Red Hat originally but now applies to all Linux
 - “raw” – files are downloaded and then a script is executed..
 - *Additional ones to be determined....*
- @password, string – the password needed to read the file if needed.
- @publisher, [string] – the name of the publisher of the file (this is obtainable from many files⁹)

If there are *multiple packages* in contents, it is assumed that they are all *logically and functionally equivalent* and that the sole difference is in the *packaging* of the contents. For example, there may be a <package type=“msi”>, <package type=“jar”> and <package type=“raw”>. The client-agent will determine which packages it is *capable of processing* and then based on preferences from the system(s) administrator; use the best-fit for the package.

Raw Program flow

³ Items like silent install, automatic acceptance of build in EULA’s are beyond the scope of OVAL because they are too specific to the environment of the system(s) administrator.

⁴ See

http://www.macrovision.com/products/flexnet_installshield/installanywhere/resources/tech_specs.shtml

⁵ See <http://en.wikipedia.org/wiki/dmg>,

http://developer.apple.com/documentation/DeveloperTools/Conceptual/SoftwareDistribution/Concepts/sd_disk_images.html

⁶ http://en.wikipedia.org/wiki/Software_Distributor

⁷ <http://www.bolthole.com/solaris/makeapackage.html>

⁸ <http://download1.sw-soft.com/Plesk/Plesk7.5/Doc/plesk-7.5r-sdk->

html/docs/plesk_modules/unix/ch11.html

⁹ See http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/sig_verification_tool.mspx?mfr=true

The oval-interpreter/client-agent will download into a temporary directory the file(s), the <script> will be written to a temporary file in this folder, and then called.

Parents:

- <contents>

Children:

- <locations> (1)
- <files> (1)
- <script> (0-1) code to do the install if needed (packages like MSI may not need this explicitly)
- <verification> (0-1) – determines if <remedy> installed successfully
- <failure> (0-1) -- actions (remedy) to do in the event of failure.

<verifications>^{vi}

A container for verifications. In general, checking for verifications is an optional behavior of clients, but it is expected that at least one verification will be checked. Many package formats support digital signatures which could eliminate the need for verification in some instances

Although the package may contain a verification, this is *internal* to the package. This presents a risk of a corrupted or infected package being substituted in some cases. To prevent this, the <remedy> and <file> must contain a ***verification mechanism*** for all packages. The same issue exists with individual files. All *external files* must be validated as that which is specified in the <remedy>

Attributes include:

- @id [required] string - in typical oval format with “sgn”
- @verificationtypes: list of verification types that are comma delimited from <verification> below.

Operation

If a @verificationuri and @verificationtypes are given, then the client may calculate the verification of the *innerXML* of the <verifications> and send it to the @verificationuri or @verificationwebservice with the @id of the <verifications>. For example

<http://verifications.patchlink.com/verify.php?id=oval:patchlink.com:sgn:1720310&md5=f15a39318cf10e8de5832da103e7fd80&&chksum=93847263>

The webservice or uri is expected to be able to handle any subset of @verificationtypes. If the verification supplied are (all) valid, then a response of “-1” is expected.

Parents:

- <file>
- <script>
- <remedy>

Children:

- <uri> - see above

- <verification > (1-n)

<verification>

This is a verification on one physical file for a piece of CDATA (in the case of scripts). Attributes include:

- @verificationtype [optional], enumeration one of
 - "chksum"¹⁰ – conforming to IEEE Std 1003.2-1992
 - "Dco"¹¹
 - "md2"
 - "md4"
 - "md5"
 - "pgp"
 - "rmd160"
 - "sha1"
 - "sha256"¹²
 - "sha384"
 - "sha512"
 - "sr160"¹³
 - "sum"
 - "sum(1)"¹⁴ - BSD and System V Unix
- Value [string] – the verification value.
- @comment [optional], string an explanation if needed

Parents:

- <verifications>

Children:

- Text

<locations>

This is the container for the possible download <uri>s. It has a single parameter

- @type [required] enumeration, one of
 - "folder" – used for "raw" packages
 - "file" – used for packages (not "raw").
- @selection [optional] enumeration, one of
 - "sequence" – use the first reachable location (default)
 - "closest" – use the closest location. The method of determining the closest is left to the client .
 - "intranet" – use the first reachable intranet location.

¹⁰ <http://man.netbsd.se/?find=cksum+1+30>

¹¹ <http://linuxdevices.com/news/NS3012318028.html> ,
http://www.osdl.org/newsroom/press_releases/2004/2004_05_24_dco.html

¹² See <http://www.netbsd.org/guide/en/chap-veriexec.html>

¹³ See http://ftp.support.compaq.com.au/pub/patches/DSNlink/dsnane300_readme.txt

¹⁴ <http://man.netbsd.se/?find=cksum+1+30>

Parents:

- <contents>

Children:

- <uri> (1-n)

<uri>

This indicates where the files may be downloaded from. In many situations, system(s) administrators may not permit download from public (internet) sites but only from intranet sites. There may also be multiple alternative sites available (some of which may be geographically closer)

- @domain [optional] string, indicates the domain that is needed to given for a logon challenge.
 - %domain% is used if this must be supplied by a system(s) administrator.
- @login [optional] string, indicates the login that is needed to be given for a logon challenge.
 - %login% is used if this must be supplied by a system(s) administrator.
- @password [optional] string, indicates the password that is needed to be given for a logon challenge.
 - %password% is used if this must be supplied by a system(s) administrator.
- Value– the URI to use to locate the files.

Parents:

- <locations>

Children:

- Text

<component>

This is a reference to another <remedy> that is part of this remedy (effectively a compound remedy).

- @remedy_ref [required] string – the id of another <remedy>
- @comment [optional], string an explanation if needed

Parents:

- <contents>

Children:

- None

<verification>

Verification consists of *post installation tests* and has children of *oval:criterion* and *oval:criteria*. It is optional (the package may raise an error that the installer is able to detect) but is usually recommended.

- @when [optional], indicates when the criterion should be evaluated, options include:

- “after install” – default
- “after reboot”
- “after logon”

Parents:

- <contents>
- <script_object>

Children:

- <criteria> (1)

<failure>

This defines what should occur if after the installation of the <remedy> the <verification> fails. Attributes are:

- @wait [optional], integer – default is 60, the number of seconds to wait before a retry.
- @retry [optional], integer – default is 1, the number of attempts to retry before stopping or trying the child <component>s.
- @comment [optional], string an explanation if needed

Parents:

- <content>

Children:

- <component> (0-n) contains a remedy_id of a remedy, typically it is an information gathering remedy.

<criteria>

This is identical to <criteria> in the <definition> node except its purpose is to verify that the <remedy> is applicable to the environment. Some examples are,

- For Windows with a Perl Script, that Perl Is installed (and an acceptable version of perl).
- For Windows with a VBS script, that security settings will permit it to run.
- Verify that the components are appropriate for the language of the operating system.

Parents:

- <oval>

Children:

- <criteria> (0-n)
- <criterion> (0-n)

<file>

A file contains information within the context of a location about a file that needs to be retrieved. The contents is the file name. It may contain the following properties:

- @relpath[optional] – the relative path from <uri>
- @verificationtype [optional], string – one of {PGP|MD5}
- @verification [optional], string

- @permissions [optional], string – in the notation that is common for the operating system, for example,
 - “ahrs” for windows : file is archived, hidden, read-only system file
 - “-r-s” for windows: file is not read only and not system
- @type [required]– enumeration
 - “binary” is application, i.e. compiled code that needs no further manipulation or decompression. The rest of the enumerations are different compressions that must be understood by the client.
 - “7z”
 - “arc”
 - “tar”
 - “zip”

Parents:

- <files>

Children:

- <verifications> (0-1)
- <name> (1)

<script>

The script is a block of CDATA that is shelled into the operating system. They are also called *shell scripts*. It may contain the following properties:

- @type – which depends on the operating system. In general it is the *extension that is needed by the operating system* to select the appropriate interpreter. The return value of the script is the <state> of
- Some possibilities are:
 - Generic {pl | py }
 - Linux – {shell}
 - Unix – {sch | tsch | sh | bash |ksh}
 - Mac – {sch | tsch | sh | bash |ksh | zsh | tcsh}
 - Windows – one of {cmd | bat | vbs | js }
 - If Perl is installed on the system then
 - If Python is installed on the system then

The value returned to the operating system may be used by<state> if it is used in a <script_test>.

Parents:

- <contents> - when used in a <remedy>
- <script_object> - when used in a <script_test>.

Children:

- <CDATA> (1) – the actual script to execute

<script_object>

This is a specific child of <objects> that contains a <script>. Scripts open up security concerns and it is recommended that <verifications> be a required child element to <script_test>.

Parents:

- <objects>

Children:

- <script> (1)
- <verifications> (1) – required for security

<script_test>

This test is a container for a <script_object>.

Parents:

- <tests>

Children:

- <object> (1) – containing a <script_object>
- <state> (0-1)
- <verifications> (1)

<policy_test>

This test queries a web service or uri (Post) to obtain *corporate policy* or similar information (corporate licensing, vendor upgrade policy etc). The <object> is the URI for a web service or a POST uri. The <state> evaluates *the single value* returned from the <object>, if there is no <state> than a value of zero is assumed to be successful.

Parents:

- <tests>

Children:

- <object> (1)
- <state> (0-1)

3.11 Import Utilities

Specific system(s) administrator information needs to be persisted in the event that new versions of solutions are issued.

The following XPath describes nodes that should be moved into any updated oval xml automatically.

- @weight for \\alternative[@id='{0}']\\criteria[@test_ref='{2}']
 - The modified @weight should be retained
- \\logic

4. Tests Status / Depreciations

As Oval evolves, different tests are depreciated and new tests are created. The table below details the current status:

Table 1 Depreciated Nodes and their replacements

Node Name	Depreciated	Status / Replacement
Debian:file_test	4.1	Unix:file_test
Debian:interface_test	4.1	Unix:interface_test
Debian:permission_test	4.1	Unix:permission_test
Debian:process_test	4.1	Unix:process_test
Debian:textfilecontent_test	4.1	Ind:textfilecontent_test
Debian:uname_test	4.1	Unix:uname_test
Debian:xmlfilecontent_test	4.1	Ind:xmlfilecontent_test
Macos:file_test	4.1	Unix:file_test
Macos:interface_test	4.1	Unix:interface_test
Macos:permission_test	4.1	Unix:permission_test
Macos:process_test	4.1	Unix:process_test
Macos:textfilecontent_test	4.1	Ind:textfilecontent_test
Macos:uname_test	4.1	Unix:uname_test
Macos:xmlfilecontent_test	4.1	Ind:xmlfilecontent_test
Oval:compound_test	4.1	Ind:compound_test
Ind:compound_test	5.0	Oval:criterion/criteria
Oval:unknown_test	4.1	Ind:unknown_test
Oval:variable_test	4.1	Ind:variable_test
Redhat:file_test	4.1	Unix:file_test
Redhat:interface_test	4.1	Unix:interface_test
Redhat:permission_test	4.1	Unix:permission_test
Redhat:process_test	4.1	Unix:process_test
Redhat:rpmversioncompare_test	4.1	Linux:rpminfo_test
Redhat:textfilecontent_test	4.1	Ind:textfilecontent_test
Redhat:uname_test	4.1	Unix:uname_test
Redhat:xmlfilecontent_test	4.1	Ind:xmlfilecontent_test
Solaris:file_test	4.1	Unix:file_test
Solaris:interface_test	4.1	Unix:interface_test
Solaris:permission_test	4.1	Unix:permission_test
Solaris:process_test	4.1	Unix:process_test
Solaris:textfilecontent_test	4.1	Ind:textfilecontent_test
Solaris:uname_test	4.1	Unix:uname_test
Solaris:xmlfilecontent_test	4.1	Ind:xmlfilecontent_test
Windows:textfilecontent_test	4.1	Ind:textfilecontent_test
Windows:xmlfilecontent_test	4.1	Ind:xmlfilecontent_test

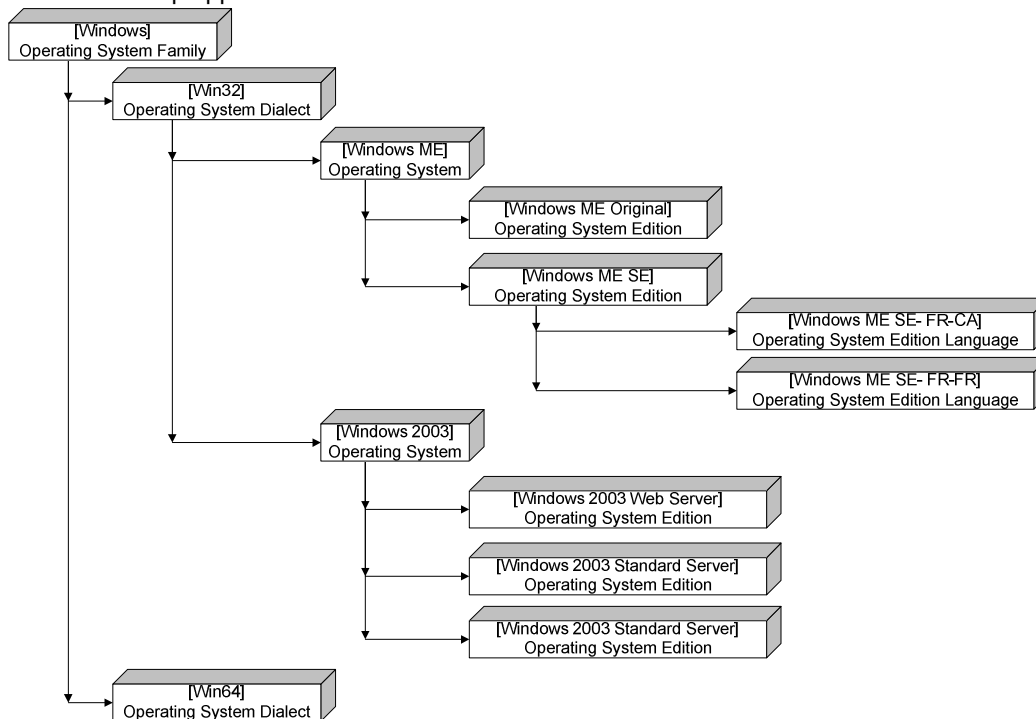
5. Product Identification

Looking forward into <remedy> and <solution> evolution, there may be a need to determine the precise Product OS and language in order to identify which remedy to use. A remedy will often be language specific. Determining which OS and which language is a functionality of <tests>, thus there is a need to define tests that will determine this. To reduce redundancy (i.e. 10 different ways to determine that you are running Windows 95), it is suggested that a <product_test> be consider which points the system / product definition that is needed.

```
<product_test id="oval:org.mitre.oval:tst:1" version="1" check="at least one"
comment="Determines in Windows 95" >
  <object object_ref="oval:org.mitre.oval:prd:1"/>
</product_test>
```

Some <remedy> may have no language components, but only the OS version.

There is a hierarchical structure of these tests which could be exploited (or at least captured). This relationship appears to be as shown below:



However there is a relational aspect. For example, Acrobat reader may be only language specific and language specific which suggests that the use of compound tests will often be appropriate. Some candidate classes are shown below.

Table 2 Scopes for <remedy> operating systems

Short Name	Description
OSF	Operating System Family
OSD	Operating System Dialect
OS	Operating System
OSE	Operating System Edition
LCID	Language
OSE-LCID	Operating System Edition for specific language
OS-LCID	Operating System for specific language
OSD-LCID	Operating System Dialect for specific language
OSF-LCID	Operating System Family for specific language

Table 3 Scopes for <remedy> products

Short Name	Description
PRDF	Product Family
PRDD	Product Dialect
PRD	Product
PRDE	Product Edition
LCID	Language
PRDE-LCID	Product Edition for specific language
PRD-LCID	Product for specific language
PRDD-LCID	Product Dialect for specific language
PRDF-LCID	Product Family for specific language

5.1 Adapting the XCCD-F Standard

With the addition of unique global identifiers in 1.1, this becomes very attractive. An issue is the method for determining which product, system it is, it would be logical to define oval tests for each item identified.

```
<oval>
  <systems>
    <system xccd-ref="whatever" xccd-ref="Windows Vista with FS">
      <criteria test_ref="oval:org.mitre.oval:tst:143239"/>
      <criteria test_ref="oval:org.mitre.oval:tst:163239"/>
    </system/>
  </systems>
</oval>
```

5.2 Adapting the CMSI Standard

The addition of @id's to the CSML definitions may be a simple flexible and quickly implemented solution. The following is an illustration (with <systems> replacing <system_list>)

```
<systems>
  <system>
    <system_part type="platform">
      <instance tag="w2k:prof"
        id="oval:org.mitre.oval:sys:1">
        <attribute_value tag="patchlevel">
```

```

    <value>SP2</value>
  </attribute_value>
</instance>
<instance tag="w2k:server"
  id="oval:org.mitre.oval:sys:2">
  <attribute_value tag="patchlevel">
    <value>SP2</value>
  </attribute_value>
</instance>
<instance tag="w2k:aserver"
  id="oval:org.mitre.oval:sys:3">
  <attribute_value tag="patchlevel">
    <value>SP</value>
  </attribute_value>
</instance>
<instance tag="w2k:dserver"
  id="oval:org.mitre.oval:sys:4">
  <attribute_value tag="patchlevel">
    <value>SP2</value>
  </attribute_value>
</instance>
</system_part>
<system_part type="software">
  <instance tag="apache"
    id="oval:org.mitre.oval:sys:5">
    <platforms>
      <platform ref_id="oval:org.mitre.oval:sys:1"/>
      <platform ref_id="oval:org.mitre.oval:sys:2"/>
      <platform ref_id="oval:org.mitre.oval:sys:3"/>
    </platforms>
    <attribute_value tag="version">
      <value>1.3.x</value>
      <value>2.x</value>
    </attribute_value>
  </instance>
</system_part>
<system_part type="software">
  <instance tag="apache_http"
    id="oval:org.mitre.oval:sys:6">
    <attribute_value tag="version">
      <value>2.x</value>
    </attribute_value>
  </instance>
</system_part>
</system>
</systems>

```

It must be noted that software can be built upon software and that not all are built on Operating Systems. For example SQL Analysis Tools is built upon SQL Server. You cannot install it without the dependency being installed.

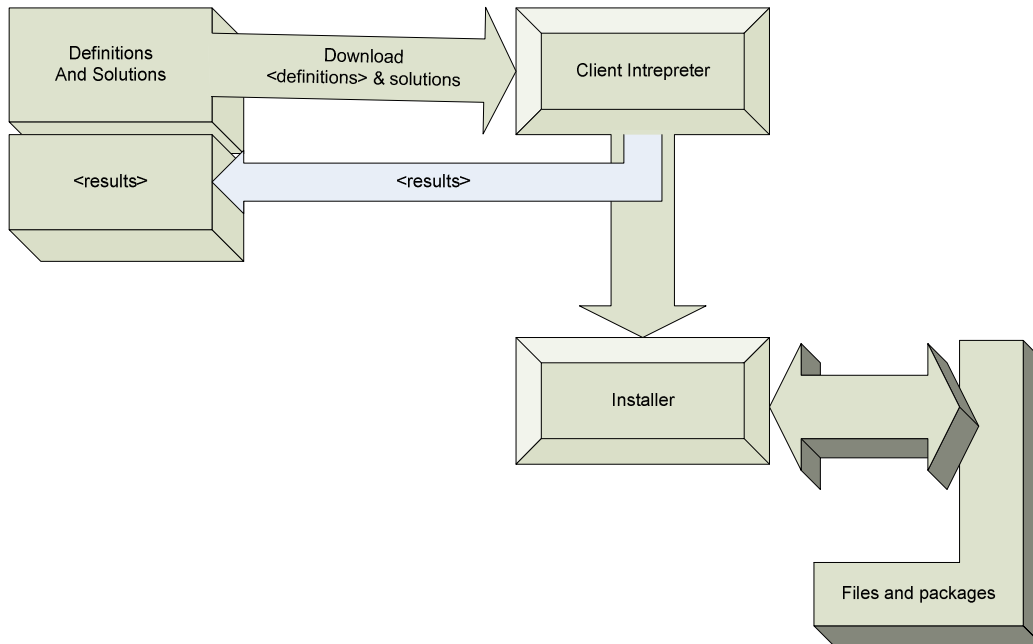
Deleted: d

Deleted:

6. General Issues

6.1 Implementation Logic:

The following diagram shows the Oval logic flow that the above would suggest:



Clients are capable of operating independently by just receiving a full oval file containing:

- <definitions>
- <tests>
- <solutions>
- <remedies>

7. Revision History

Version	Date	Author(s)	Description
1.0	2005-06-10	Ken Lassenen	<ul style="list-style-type: none"> Initial Version
1.0.1	2005-07-08	Ken Lassenen	<ul style="list-style-type: none"> First Draft for Oval Developer Days

8. Bibliography

- John Dunagan, Roussi Roussev, Brad Daniels, Aaron Chad Verbowski, Yi-Min Wang **Towards A Self-Managing Software Patching Process Using Black-Box Persistent-State Manifests**
- Neal Ziring, **Specification for the Extensible Configuration Checklist Description Format Platform Facts Definition (XCCDF-P)** <http://checklists.nist.gov/docs/xccdf-spec-1.1.pdf>
- Bernd Grobauer **CVE, CME, . . . , CMSI? – Standardizing System Information** <http://www.first.org/conference/2005/papers/dr.-bernd-grobauer-paper-1.pdf>
- Microsoft, **Authenticode Signing for Game Developers**, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/Authenticode_Signing_for_Game_Developers.asp
- Sun, **Signing and Verifying JAR Files**, <http://java.sun.com/docs/books/tutorial/deployment/jar/signindex.html>

9. Notes on Taxonomy of Terms

Formatted: Bullets and Numbering

ⁱ Synonyms for <solution> were examined with the following conclusions:

- answer – too many meanings
- result -- this is a post-solution state
- resolution -- this is a post-solution state
- solvent -- unfortunately has an additional meaning in chemistry, word confusion risk

ⁱⁱ Synonyms for <remedy> were examined with the following conclusions:

- Correct – too many meanings
- Correction – too many meanings
- Cure – implies making healthy, a remedy may be an amputation which is not a cure
- Redress – implies a wrong
- Curative – a verb, should be a noun
- Rectify – obtuse
- Repair – see cure above
- Amend – see cure above, implies leaving intact which may not always be the case.

Deleted: , implies

ⁱⁱⁱ Synonyms for @weight were examined with the following conclusions:

- Importance – implies a categorical weight
- Influence – a bit fuzzy...
- Priority – implies a resolved ordering
- Precedence -- implies a categorical weight
- Order – implies an ordering
- Sequence -- implies a resolved ordering

^{iv} Synonyms for <alternative> were examined with the following conclusions:

- Protocol – a standard medical term but is used commonly for communications.
- Choice – not specific enough
- Option – not specific enough

^v The term “remediation” is used in the literature, so remedy appears appropriate.

^{vi} The following terms were considered:

- Signature: sometimes s called a “magic number”, can also be applied to a method as a way of *identification*. (Its synonyms are: name, autograph, cross, sign, moniker, monogram, mark etc).
- Confirmation – also applies to transmission of packets as well as user confirmation
- Corroboration – not enough sense of “certainty”
- Proof – not clear enough
- Authentication – usually applied to users and not files
- Certification – usually applies to user certificates and implies an issuing authority
- Substantiation – does not have enough sense of certainty.

Deleted: s

Deleted: it's

Deleted: certainty

Deleted: y

Deleted: certainty