Abstract

The DMTF Common Information Model (CIM) is a conceptual information model for describing computing and business entities in enterprise and Internet environments. It provides a consistent definition and structure of data, using object-oriented techniques. The CIM Schema establishes a common conceptual framework that describes the managed environment.

The CIM Application Management Model is an information model that describes the details commonly required to manage software products and applications. This model can describe applications with various structures – ranging from standalone desktop applications to a sophisticated, multi-platform distributed, Internet-based application. Likewise, the model can be used to describe a single software product as well as a group of interdependent software products that form a business system.
Notices

DSP0140 \hspace{1cm} Status: Preliminary

Copyright © 1998-2003 Distributed Management Task Force, Inc. (DMTF). All rights reserved.

DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

Implementation of certain elements of this standard or proposed standard may be subject to third party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights, and is not responsible to recognize, disclose, or identify any or all such third party patent right, owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third party patent rights, or for such party’s reliance on the standard or incorporation thereof in its product, protocols or testing procedures. DMTF shall have no liability to any party implementing such standard, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

For information about patents held by third-parties which have notified the DMTF that, in their opinion, such patent may relate to or impact implementations of DMTF standards, visit http://www.dmtf.org/about/policies/disclosures.php.
# Table of Contents

Abstract ............................................................................................................................... 1  
Notices ................................................................................................................................ 2  
Table of Contents ................................................................................................................ 3  
1. Introduction ..................................................................................................................... 4  
2. Basic Concepts ................................................................................................................ 6  
   2.1 Software Elements Details ........................................................................................ 8  
      2.1.1 Software Element Life Cycle or State ............................................................... 8  
      2.1.2 Software Element Conditions ........................................................................ 9  
      2.1.3 Software Element Actions ........................................................................... 11  
3. Application Management Model Use Case ................................................................. 14  
   3.1 Product or Application System ............................................................................... 14  
   3.2 Software Feature or Software Elements ................................................................. 15  
   3.3 Decomposing Software Features into Software Elements ..................................... 17  
   3.4 Capturing Software Element Conditions ............................................................... 18  
   3.5 Decomposing Software Elements ........................................................................ 21  
   3.6 Discovery .............................................................................................................. 25  
   3.7 Shared Software Elements ................................................................................... 26  
   3.8 Dependencies ....................................................................................................... 28  
   3.9 Application System Details .................................................................................. 29  
Acknowledgments ......................................................................................................... 29  
Appendix A – Change History .......................................................................................... 30  
Appendix B – List of Files for Full-Featured Installation ................................................ 30  
   Oracle 8 File List ......................................................................................................... 34  
   CustomerFirst Management Object Format (MOF) .................................................... 39  
   Application Management Object Format (MOF) ....................................................... 49
1. Introduction

The CIM Application Management Model is an information model that describes the details commonly required to manage software products and applications. Figure 1 summarizes some of the ways this information model can be used.

In most cases, developers start with application/software providers, producing instances of the classes in the model to describe the manageable aspects of their product. The manageable aspects include details such as installable or deployable units, the files these units are composed of, the memory required to properly execute the software, etc. These providers can either become familiar with the details of the information model, so they can produce the details by hand, or application development tool vendors can enhance their tools to produce the information based on this open standard. As an example of the latter, a software change management tool typically knows the files that are produced from a build and can automatically record this information in the CIM application management model format.

By packaging an instance of the application management information model with an application or other software, operating systems utilities and/or management tools can consume the model in order to simply the work of their users. In the case of operating systems, utilities that install software or maintain databases of installed software can consume the information model, so users do not need to type the details in. On the other hand, management tools have instances of the information model to eliminate the need for users to type in details. For example, a software distribution tool can learn what needs to be done when the software is distributed to various operating system platforms. So, the information model can be used by a wide variety of tools.
Version 2 of the CIM Application Management Model is not expected to capture all the information these various tools require to accomplish their missions. The model provides a base upon which additional modeling concepts can be added. For example, management tools that monitor or measure applications can use the basic concepts of this model to describe the components of a software product that can be monitored, but details of how these components can be measured (e.g., output queue length) must be added. Similarly, management tools that deal with software maintenance issues can use the Version 2 model to capture the details of complete products, but must add additional modeling constructs to capture fixes or patches to be applied to this base.

In addition to the basic model application model lifecycle components, a special category of classes related to BIOS software which inherits from the application model are now included in the application model. This includes classes such as CIM_BiosFeature which is subclassed from CIM_SoftwareFeature. These components are really part of the Bios model and originally existed within the System devices group and SYSDEV model. They are now part of the application model mof because they use the application model classes as superclasses and therefore are only usable if the application model is installed. Please see the System model for more information on the use of these classes.
2. Basic Concepts

The application management model uses the concepts of a software product, software feature, software element, and application system. These concepts are defined as follows:

- A **Software Product** is a collection of software features that can be acquired as a unit. Acquisition implies an agreement between the consumer and supplier, which may have implications in terms of licensing, support, or warrantee.
- A **Software Feature** is a collection of software elements that performs a particular function or role of a software product. This level of granularity is intended to be meaningful to a consumer or user of the application to choose. This concept allows software products or application systems to be decomposed into units that have a meaning to users rather than units that reflect how the product or application was built (i.e., software elements).
- A **Software Element** is a collection of one or more files and associated details that are individually managed on a particular platform. It represents the level of granularity at which software features are managed.
- An **Application System** is a collection of software features that can be managed as an independent unit that supports a particular business function.

As the description indicates, the concepts of software product, application system, and software feature group or organize software elements. Note that a **Software Element** represents the level of granularity that is the fundamental building block of the application management information model.

These four concepts and the relationships between them are represented in the information using the classes and associations shown in Figure 2.
A software product is defined using the Product class in the CIM Core Model. This class is a collection of software features that can be acquired as a unit. Acquisition implies an agreement between the consumer and supplier, which may have implications in terms of licensing, support, or warranties.

Also noted above, Application System is a subclass of the System Model. The software element and software feature concepts are subclasses of Logical Element, known as SoftwareElement and SoftwareFeature.
2.1 Software Elements Details

The model uses three additional concepts to organize more details about software elements. These include:

- software element life cycle or state
- environmental conditions that software elements depend on, and
- software element actions.

Figure 3 shows a conceptual picture of how these fit together. The next three sections provide a detailed description.

![Software Element States, Conditions, and Actions](image)

2.1.1 Software Element Life Cycle or State

The most basic aspect of managing software elements is managing its transitions through its life cycle. The life cycle can be segmented into four activities:

1. deploying a software feature
2. installing and configuring a software feature
3. starting a software feature
4. monitoring and operating a running software feature

As the software performs these tasks, the details of a particular software element changes. For example, the files that make up a software element when it is deployed are different from the files that make up a software element when it is running on a machine. Therefore, managing a software element through the act of performing these four tasks requires a management tool to understand what the particular details are before and after the tasks are performed. Using the value of the SoftwareElementState property in the SoftwareElement class, the following four states can be captured:
• A software element in the **deployable state** describes the details necessary to successfully distribute it, as well as the details (conditions and actions) required to create a software element in the installable state (i.e., the next state).

• A software element in the **installable state** describes the details necessary to successfully install it, as well as the details (conditions and actions) required to create a software element in the executable state (i.e., the next state).

• A software element in the **executable state** describes the details necessary to successfully start it, and the details (conditions and actions) required to create a software element in the running state (i.e., the next state).

• A software element in the **running state** describes the details necessary to monitor and operate on a start element.

### 2.1.2 Software Element Conditions

Conditions describe situations that may or may not exist in the environment where a software element also may or may not exist. The environment is typically a computer system. Conditions are organized into two groups: in-state conditions and next-state conditions.

The **in-state conditions** describe the characteristics of an environment that contains a software element in a particular state. Some examples of these characteristics are the directories that should exist, as well as the files that should be in these directories. For an existing software element, in-state conditions can be used to verify that the software element is complete or whole.

The **next-state conditions** describe the minimum set of characteristics that must be true in the target environment for the next-state actions (See Section 2.1.3 Software Element Actions) to successfully execute. The next-state conditions are the minimum pre-conditions required for the next-state actions to be successful.

Figure 4 shows the Application Model constructs that model both in-state and next-state conditions. The Check class is used to represent a condition, and the SoftwareElementChecks association links particular conditions with the appropriate Software Element. The value of the Phase property on the SoftwareElementChecks association determines whether a particular instance of a Check class is an in-state or next-state condition. The details of a particular condition (e.g., expected amount of disk space) are captured in a subclass of the Check class. The conditions covered in Version 2 of the Application Management Model and their interpretation is summarized in Table 1.
### Condition	In-State Interpretation	Next-State Interpretation
---
**Memory Requirements**	Minimum Amount of memory required to transition into the *current* state.	Minimum amount of memory required to transition into the *next* state.

**Disk Space**	Minimum amount of disk space required to transition into the *current* state.	Minimum amount of disk space required to transition into the *next* state.

**Swap Space**	Minimum amount of swap space required to transition into the *current* state.	Minimum amount of swap space required to transition into the *next* state.

**Architecture**	The architecture required by a software element in the *current* state.	The architecture required by the software element to transition into the *next* state.

**Files**	A file that is expected to exist or not exist when a software element is in the *current* state.	A file that is expected to exist or not exist before a software element transitions into the *next* state.
### Table 1. Conditions (the "Check" Class) for Application Deployment

<table>
<thead>
<tr>
<th><strong>Directories</strong></th>
<th>A directory that is expected to exist or not exist when a software element is in the current state.</th>
<th>A directory that is expected to exist or not exist before a software element transitions into the next state.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS Version</strong></td>
<td>The version or ranges of versions a software element requires in its current state.</td>
<td>The version or ranges of versions a software element requires before it transitions into the next state.</td>
</tr>
<tr>
<td><strong>Software Elements</strong></td>
<td>A software element that is expected to exist or not exist when a software element is in the current state.</td>
<td>A software element that is expected to exist or not exist before a software element transitions into the next state.</td>
</tr>
<tr>
<td><strong>Setting</strong></td>
<td>An entry in a &quot;setting&quot; file that is expected to exist or not exist when a software element is in the current state.</td>
<td>An entry in a &quot;setting&quot; file that is expected to exist or not exist when a software element is in the next state.</td>
</tr>
<tr>
<td><strong>Version Compatibility</strong></td>
<td>Indicates whether it is permissible to create the software element in its next state.</td>
<td></td>
</tr>
</tbody>
</table>

The model does not require all the aspects to be consistent. For example, there can be an in-state condition for one file to exist, and there can be a next-state condition for a different file to exist. In this case, the next-state condition says the files must be there before executing the next-state actions, where the in-state condition says this file should be there after the software element in the new state is created.

A user of the model cannot assume that the difference between the in-state conditions and the previous state’s in-state condition will equal the sequence of actions. There may or may not have been an action to copy/move the file, depending on whether it is an explicit action or a side effect of executing another action (e.g., running setup.exe).

An important special case deals with the Runnable state. Since what follows the running state is the executable state, knowing uninstall actions for a software element in the running state has no meaning. The next-state conditions of a software element in a runnable state captures the details that lead to a successful shut down.

### 2.1.3 Software Element Actions

Actions are operations that either create a new software element or remove an existing software element. Actions are organized into two categories: next-state actions and uninstall actions.
The **next-state actions** describe a sequence of actions that will create a software element in the next state using a software element in the current state. These are the actions that perform the task described earlier (unpackage, install, start). The next-state actions do not destroy the software element in the previous state. The next-state conditions are pre-conditions that need to be true about the environment in order for this sequence of actions to succeed. These are not the only preconditions, since the in-state conditions can also be thought of as preconditions — the next-state actions assume that a software element exists in the current state.

The **uninstall actions** describe a sequence of actions that will properly remove a software element in a particular state from a machine. This is commonly used to uninstall software elements. The in-state conditions can be thought of as pre-conditions for the uninstall action. As shown in Figure 3, the uninstall actions remove or delete software elements. They do not create a software element in the previous state.

The class and associations used to capture action sequences in the Application Model are shown in Figure 5. The Action class represents a particular action. The value of the Direction property in the Action class determines whether the action is a next-state action or an uninstall action. SoftwareElementActions associations link the actions to a particular software element. The ActionSequence association is used to order the actions for a particular software element when there is more than one. The details of a particular type of action are defined in a set of subclasses. Table 2 summarizes the action defined in the version 2.0 model.

![Figure 5. Classes and Associations for Actions](image-url)
<table>
<thead>
<tr>
<th>Actions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Directory</td>
<td>An action to create or remove a directory.</td>
</tr>
<tr>
<td>File</td>
<td>An action to create or remove a file.</td>
</tr>
<tr>
<td>Reboot</td>
<td>An action that signals the need to reboot the computer system.</td>
</tr>
<tr>
<td>Execute</td>
<td>An action that executes a program.</td>
</tr>
<tr>
<td>Program</td>
<td>This can be the install script or program (e.g., setup.exe) when a software element in the installable state transitions to the executable state.</td>
</tr>
<tr>
<td>Modify Setting</td>
<td>An action to modify an entry in a &quot;setting&quot; file.</td>
</tr>
</tbody>
</table>

Table 2. Actions Defined in Application Deployment

<table>
<thead>
<tr>
<th>State</th>
<th>Subpart</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployable</td>
<td>Next-State Actions</td>
<td>These actions perform the “unpackage” task to create a software element in the installable state.</td>
</tr>
<tr>
<td></td>
<td>Uninstall Actions</td>
<td>These actions remove the software element in the deployable state.</td>
</tr>
<tr>
<td>Installable</td>
<td>Next-State Actions</td>
<td>These actions perform the install task to create a software element in the executable state.</td>
</tr>
<tr>
<td></td>
<td>Uninstall Actions</td>
<td>These actions remove the software element in that state.</td>
</tr>
<tr>
<td>Executable</td>
<td>Next-State Actions</td>
<td>These actions start the software element.</td>
</tr>
<tr>
<td></td>
<td>Uninstall Actions</td>
<td>These actions uninstall the executable state.</td>
</tr>
<tr>
<td>Runnable</td>
<td>Next-State Actions</td>
<td>These actions stop or terminate the running software element.</td>
</tr>
<tr>
<td></td>
<td>Uninstall Actions</td>
<td>These are not typically used for a software element in a runnable state.</td>
</tr>
</tbody>
</table>

Table 3. "Change State" Actions Defined in Application Deployment
3. Application Management Model Use Case

This section illustrates how to use the CIM Application Management model for the fictional CustomerFirst software product. This is a customer support and problem tracking application developed by the First Always Company. Typical CustomerFirst users are technical support staff in a computer-related products company. They rely on CustomerFirst to log customer call incidents, research resolutions, and assign and track ownership of problems.

CustomerFirst is a client/server application with a thick client and a database server configuration. The clients run on various MS-Windows platforms and communicate with relational databases running on servers. Since companies typically have some users tracking customer calls or incidents, and have another set of users managing the resolution of problems, the First Always Company ships their products with two different client configurations: an Incident Manager client and a Problem Tracker client.

The CustomerFirst product uses other products like an Oracle Relational DBMS to deliver its functionality. In addition, Customer First can deliver additional capabilities when a customer has additional software products. These include:

1. The Problem Tracker client can use external e-mail services to notify its users of events. It supports a variety of mail systems and mail protocols.
2. Users with Microsoft Excel can create graphs and charts by accessing data from the CustomerFirst database. This feature is distributed and installed as an add-in to Excel.

In the following sections, we present a step-by-step discussion to create the appropriate Application Management model objects for CustomerFirst. When more details about the product are needed for the model (for example, memory and disk space requirements), this will be introduced.

3.1 Product or Application System

CustomerFirst is a software product purchased from the First Always Company. The CustomerFirst application uses other software products, like Oracle, to accomplish its tasks. Therefore, a Product class needs to be created for the CustomerFirst product. A graphical representation of the Product object and the matching MOF statement are:
Table 4. CustomerFirst Product

Since the CustomerFirst product uses other software products, developers can combine its components with these products to form an application system. The section titled Application System Details illustrates this concept for the WeServiceIt company.

3.2 Software Feature or Software Elements

The Product object captures high-level details about the vendor of the product. The next step is to decompose the CustomerFirst product into its software features. From the overview of the CustomerFirst product we derive the following three features:

1. Incident Manager
2. Problem Tracker
3. Advance Graphics Package

For each of these, the developer must create a SoftwareFeature object, and these objects must be associated with the Product object using the ProductSoftwareFeature association. These associations are required because SoftwareFeature objects are named within the context of Product objects, meaning SoftwareFeature objects are weak with respect to Product objects. A graphical representation of the software features is shown in Figure 6.
Table 5. CustomerFirst SoftwareFeature MOF

The values for Vendor, ProductName, ProductNumber, IdentificationNumber, and Version are propagated keys from the CIM_Product object referenced by the CIM_ProductSoftwareFeatures association.

Figure 6. CustomerFirst Software Features
3.3 Decomposing Software Features into Software Elements

Software Features are decomposed into manageable units known as software elements. The software elements for a particular software feature are organized by two dimensions: platform and state. A particular Software Feature can have one or more software element for each platform and state combination. Table 6 summarizes the eight software elements needed for the Incident Management features. This software feature has a single software element for each of the platform/state combinations.

<table>
<thead>
<tr>
<th>Platform</th>
<th>Deployable</th>
<th>Installable</th>
<th>Executable</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>NT</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Windows 95</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 6. Software Elements for Incident Management Feature

Figure 7 shows a UML instance diagram for the four software elements defined for the NT platform. There is one SoftwareElement object for each software element.

The MOF used to define and link one of these software elements (i.e., the installable state element) to the Incident Management feature is shown in Table 7. The MOF for the other
three software elements would be similar, except for the Name and the SoftwareElementState properties.

```plaintext
instance of CIM_SoftwareElement as $elmt001_01
{
   Name = "Incident Manager Element"; // key
   TargetOperatingSystem = 17; // NT
   Version = "01.01.01"; // key
   SoftwareElementState = 0; // Deployable
   SoftwareElementID = "Definition"; // key
   Version = "01.01.01"; // key
   Manufacturer = "First Always, Co.";
   SerialNumber = "";
   LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
   GroupComponent = $feat001;
   PartComponent = $elmt001_01;
};
```

Table 7. MOF Example of a Software Element Linked to a Software Feature

### 3.4 Capturing Software Element Conditions

For each software element, the Application Management model allows more details about items like disk space requirements, operating system version, etc. For example, Table 8 below shows the details of the Software Elements for the Incident Management software feature. The table captures some of the details that can be specified for each software element. These details include the version of the OS, the amount of disk space required to transition into the next state, and the amount of memory required for a software element within a particular state.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Platform</th>
<th>Details</th>
<th>Deployable</th>
<th>Installable</th>
<th>Executable</th>
<th>Running</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incident Manager</td>
<td>Win. 95</td>
<td>Version</td>
<td>3.5.1</td>
<td>3.5.1</td>
<td>3.5.1</td>
<td>3.5.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Disk Space</td>
<td>.75 MB</td>
<td>2 MB</td>
<td>15 MB</td>
<td>3 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Memory</td>
<td></td>
<td></td>
<td>8 MB</td>
<td>32 MB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software</td>
<td></td>
<td></td>
<td>Oracle</td>
<td>Oracle</td>
</tr>
</tbody>
</table>
Table 8. Incident Management Feature Condition Summary

These details are captured using Check objects. Recall that checks or conditions are organized into two lists: in-state conditions and next-state conditions. This distinction must be considered when mapping the details summarized in Table 8. According to the definitions in Table 1, the details can be interpreted as in-state conditions, so:

- the version row refers to the operating system version required by the software element in the specified state,
- the disk space row refers to the minimum amount of disk space required to transition in the specified state,
- the memory row refers to the minimum amount of memory required to transition into the specified state,
- the software row refers to the expected software element(s) that is expect to exist for the software element in the specified state.
It is not necessary, but it is reasonable to consider defining next-state conditions for conditions such as version, disk space, and memory on the previous state. For example, the 15 MB of disk space required to transition the Incident Manager on NT software element into its executable state can also be specified as a next-state condition for the same software element in the installable state. Figure 8 shows the Check objects required for the Incident Manager software element targeted for NT in the executable state using for the OS Version details. Similar objects need to be created for disk space and memory.

![Figure 8. Check Object for NT/Executable Software Element](image)

The following is the MOF for the two OSVersionCheck classes. Remember that the Phase property of the SoftwareElementChecks association is what determines whether a check is an in-state or next-state condition.

```mo
instance of CIM_OSVersionCheck as $chck001_02_N01
{
    Name = "Incident Manager Element"; // key
    TargetOperatingSystem = 17;
    Version = "01.01.01";
    SoftwareElementState = 1;
    SoftwareElementID = "Definition"
    Manufature = "First Always, Co,"
    SerialNumber = "";
    LanguageEdition = "en"
    Caption = "";
    CheckMode = True
    MinimumVersion = "3.5.1"
    MaximumVersion = "3.5.1"
}
```

DSP140 20
Table 9. Example SoftwareElementChecks

3.5 Decomposing Software Elements

The application management model allows details about the software element’s bill of materials - a list of all files that make up the component – including executables, data files, scripts, and so on. These files are organized according to the target directory where they will reside in the target environment or system.

We model two aspects of the bill of materials: the directory structure of the software element using the DirectorySpecification class, and the file content of these directories using the FileSpecification class. The software element files are modeled in the context of
a directory structure so that it is easier to relocate these files. Rather than changing the path of every file, only the path for the directory must be changed.

The DirectorySpecification and FileSpecification classes are subclasses of Check, since they described conditions that can and must exist within the computer system environment.

The DirectorySpecification class has three properties: Index, Location Type, and Path. The Index property is a unique number assigned to a particular directory. The Location Type property describes the type of directory. The possible directory types are summarized in Table 10. The Path property allows an application provider to describe a default or recommended path for a particular directory. The value can be changed for a particular building block on a particular system.

<table>
<thead>
<tr>
<th>Product Directories</th>
<th>Shared Directories</th>
<th>System Directories</th>
<th>Miscellaneous Directories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product base directory</td>
<td>Shared base directory</td>
<td>System base directory</td>
<td>Unknown</td>
</tr>
<tr>
<td>Product executables directory</td>
<td>Shared executables directory</td>
<td>System executables directory</td>
<td>Other</td>
</tr>
<tr>
<td>Product library directory</td>
<td>Shared library directory</td>
<td>System library directory</td>
<td></td>
</tr>
<tr>
<td>Product include directory</td>
<td>Share include directory</td>
<td>System include directory</td>
<td></td>
</tr>
<tr>
<td>Product configuration directory</td>
<td></td>
<td>System configuration directory</td>
<td></td>
</tr>
<tr>
<td>Product log directory</td>
<td></td>
<td>System log directory</td>
<td></td>
</tr>
<tr>
<td>Product working directory</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 10. Directory Types for the Location Type Property

The files that are in a particular target directory are captured using the FileSpecification class and are referenced by the DirectorySpecificationFile association.

The following are files prepared by Installshield, ready to be copied to a distribution CD-ROM. This represents the software element for the Incident Management Feature in its Installable state.
This list would be modeled as a single directory with these 19 files, as shown in the Figure 9.
instance of CIM_SoftwareElementChecks
{
    Element =
    
    Name = "IM. On NT in Depl. State",
    TargetOperatingSystem = 17,
    Version = "01.01.01",
    SoftwareElementState = 0,
    SoftwareElementID = " Definition",
    Version = "01.01.01"
    
    Check =
    Name = " PT Primary Element",
    TargetOperatingSystem = 17,
    Version = "01.01.01",
    SoftwareElementState = 0,
    SoftwareElementID = "Definition",
    Version = "01.01.01",
    CheckID = "OS VersionCheck"
    
    Phase = 0; // in-state
};

instance of CIM_DirectorySpecification as $dirspec_01
{
    Name = "IM. On NT in Depl. State"; // key
    TargetOperatingSystem = 17; // NT
    Version = "01.01.01"; // key
    SoftwareElementState = 0; // Deployable
    SoftwareElementID = " Definition"; // key
    Version = "01.01.01"; // key
    CheckID = "OS VersionCheck"
    Description = " ";
    Caption " ";
    CheckMode = True;

    DirectoryType =??;
    DirectoryPath = " ";
};

instance of CIM_FileSpecification as $filespec_01
{
    Name = " ";
    CreateTimeStamp = "xx/xx/xx";
    FileSize=123; //kilobytes
    CheckSum = 00000;
    CRC1 = 0000;
    CRC2 = 0000;
    MD5Checksum = "1234567890123456";
};
Table 11. MOF describing a file list modeled as a single Directory

Although the model allows an application provider to capture directories and files in ways independent of physical location, there are situations in which the application provider will want to "hard code" the physical location of a file. This can be accomplished in the model by providing the complete file name in the file specification.

Table 12. CustomerFirst Client-side Features

<table>
<thead>
<tr>
<th>Component (Features)</th>
<th>Dependency</th>
<th>Main Executables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Advance GraphicsPackage</td>
<td>Requires Microsoft Excel</td>
<td>CFSRPT.XLA</td>
</tr>
<tr>
<td>part of</td>
<td>Requires Problem Tracker</td>
<td>(installed as Excel add-in)</td>
</tr>
<tr>
<td></td>
<td>feature</td>
<td>DYNMCSQL.DLL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LOADQDB.DLL</td>
</tr>
</tbody>
</table>

3.6 Discovery

Sometimes, a developer must discover or detect software components on a set of machines. Developers typically accomplish this using distinguishing characteristics of critical files that a software component must have installed on the target system to function. These characteristics are referred to as a software signature. The file size, file time stamp, file checksum, CRC1 and CRC2 properties of the FileSpecification class form the software signature defined by the DMTF Software Standard Group Definition, Version 2.0.

The CRC 1 property is determined by running an algorithm on a particular 512K bytes of the file. The CRC algorithm is specified by the ITU recommendation V.41.

For the CRC 1 value, the algorithm is run on the middle 512K bytes of the file. To determine the middle 512K bytes, do the following:
1. Identify the file's midpoint by dividing the file size by two and rounding down to the next whole number.
2. Take the 256K bytes on either side of the file's midpoint. The resulting 512K bytes is the portion of the file on which you should run the algorithm.

If the file is less than 512K bytes, then run the algorithm on the entire file.

For the CRC 2 value, the algorithm is run on the 512K bytes that result from the following procedure:

1. Divide the file by three and save the remainder.
2. Add the remainder to the beginning of the file (starting from zero). The result is the midpoint that is used to establish the 512K bytes.
3. Take the 256K bytes on either side of the midpoint determined in step 2. The resulting 512K bytes is the portion of the file on which you should run the algorithm.

If the file is less than 512K bytes, then run the algorithm on the entire file.

### 3.7 Shared Software Elements

After closer scrutiny, we determined that the executable software elements have shared files, so a shareable feature can be created.

The *CustomerFirst* application is mostly represented by components executing on the client (desktop) system. Typical setup is only one copy of the runtime components on a LAN shareable directory. All clients systems load the software from the shared area when *CustomerFirst* or its features are invoked.

Observe that some software elements are shared across software features. You can determine this from the model by locating software elements that reference more than one Software Feature.
<table>
<thead>
<tr>
<th>Software Element</th>
<th>Dependency</th>
<th>Main Executables</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Functions including Database Access</td>
<td>Oracle Call Interface; Windows 95/NT</td>
<td>CSMDBIO.DLL</td>
<td>CustomerFirst installation disks</td>
</tr>
<tr>
<td>CustomerFirst call tracking and defect tracking</td>
<td>Core Functions; Windows 95/NT</td>
<td>CSF.EXE</td>
<td>CustomerFirst installation disks</td>
</tr>
<tr>
<td>(consists of over 300 .dll’s / the main executable loads the appropriate .dll as the feature is invoked. Each .dll represents a main UI panel and subordinate panels and dialog boxes.)</td>
<td></td>
<td>INCIDENT.DLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROBLEM.DLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INCSERCH.DLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomerFirst e-mail facilities</td>
<td>CustomerFirst call tracking; one of the following client components:</td>
<td>RTIMAPI.DLL</td>
<td>CustomerFirst installation disks</td>
</tr>
<tr>
<td></td>
<td>• MAPI-compliant mail software such as Microsoft Mail; Microsoft Exchange; Windows Messaging; Microsoft Outlook</td>
<td>VIMMAIL.DLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• VIM-compliant mail software such as Lotus CC:Mail</td>
<td>LOTMAIL.DLL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Lotus Notes Mail</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CustomerFirst e-mail facilities (SMTP mail)</td>
<td>CustomerFirst call tracking; SMTP mail server</td>
<td>SMTPMAIL.DLL</td>
<td>CustomerFirst installation disks</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 13. Shared Software Elements

<table>
<thead>
<tr>
<th>Client Component</th>
<th>Dependency</th>
<th>Main Executables</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>VIM-compliant mail (CC:Mail)</td>
<td>Windows; supported network protocol such as IP.</td>
<td>??</td>
<td>Lotus CC:Mail CD or CustomerFirst installation disks.</td>
</tr>
<tr>
<td>MAPI-compliant mail software</td>
<td>E-Mail server supporting the MAPI software</td>
<td>??</td>
<td>E-mail software from the appropriate vendor.</td>
</tr>
<tr>
<td>Oracle Call Interface</td>
<td>Oracle IP Protocol Adapter</td>
<td>??</td>
<td>Oracle Client Software CD</td>
</tr>
<tr>
<td>Oracle IP Protocol Adapter</td>
<td>IP stack provided by Windows; IP network hardware</td>
<td>??</td>
<td>Oracle Client Software CD</td>
</tr>
</tbody>
</table>

Table 14. Client Components

3.8 Dependencies

We have learned how to describe dependencies that a software element might have on an operating system, such as amount of memory, disk space, etc. The application management information model also provides mechanisms to express higher-level dependencies. We can use the Advance Graphics Package feature to illustrate some of
these. In Table 12, the Advance Graphics Package feature has two dependencies: it requires Microsoft Excel and it requires the Problem Tracker feature. These represent inter-component or inter-feature dependencies. When application management models are available for both, a developer can use an instance of the SoftwareFeatureCheck class. However, in many cases an application model may not be available, as in the case of Microsoft Excel in this example. In this case, you can express dependencies using the DirectorySpecification and FileSpecification classes to locate one or more significant files for the Microsoft Excel product.

<table>
<thead>
<tr>
<th>TBD</th>
</tr>
</thead>
</table>

### Notes

1. The values for Vendor, ProductName, ProductNumber, IdentificationNumber, and Version are propagated keys from the CIM_Product object referenced by the CIM_ProductSoftwareFeatures association.

<table>
<thead>
<tr>
<th>Table 15. Dependencies expressed by Directory and File Spec Checks</th>
</tr>
</thead>
</table>

### 3.9 Application System Details

The CustomerFirst product might use several other products. These include a relational database product, an e-mail system, and Microsoft’s Excel. Each of them would have a similar CIM_Product.

Instance diagram and MOF TBD

### Acknowledgments

The Application Management Work Group’s ability to develop this model was greatly enhanced by work that already existed in the industry. We would like to recognize these sources since they served as a source of ideas and provided experiences that separate concepts from practical implementation. These sources include the Posix 13786 standard, the DMTF’s Software Standard Group Definitions, Microsoft’s software and installation work for Windows 2000, and Tivoli’s Application Management Specification Version 2.0.
Appendix A – Change History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>May 17, 1998</td>
<td>Initial Version</td>
</tr>
<tr>
<td>1.1</td>
<td>May 25, 2002</td>
<td>Update to reflect changes to the model effective CIM 2.6. Note that changes (See the application MOF for details) are not reflected in the model descriptions in this version of this document. Also removed the section which was to have contained the complete Application Model UML.</td>
</tr>
<tr>
<td>1.2</td>
<td>June 12, 2003</td>
<td>Update Document to CIM 2.7 Clean up graphics and text. Add comments on the new components of the model (bios) and the relationship between the model and identity.</td>
</tr>
</tbody>
</table>

Appendix B – List of Files for Full-Featured Installation

The following is a list of files of a full-featured installation of CustomerFirst. This represents the application in its executable state.

```
01/01/80 12:00a <DIR> .
01/01/80 12:00a <DIR> ..
01/30/98 10:43a 82,944 PRODPGS.DLL
03/24/98 07:48p 5,632 RTIOBJ.DLL
01/30/98 10:36a 43,008 INCOMORD.DLL
01/30/98 11:05a 52,736 MSRCRES.DLL
01/30/98 02:16p 26,112 RTIEDIT.DLL
03/24/98 08:04p 129,024 ADM.EXE
01/09/98 02:57p 8,192 CSMUSER.DLL
03/24/98 09:09p 4,608 MHSHIELD.DLL
03/24/98 07:48p 5,632 RTIOUT.DLL
02/15/95 01:11a 17,920 IMPLODE.DLL
08/24/96 11:11a 326,656 MSVCRT40.DLL
07/19/96 04:25p 267,536 MSVCRT.DLL
01/09/98 02:52p 4,096 RTIDL.G.DLL
01/09/98 04:09p 3,507,458 SYSADMIN.HLP
01/30/98 10:47a 28,672 R0008.DLL
01/10/98 04:48p 8,249,536 CFS.HLP
03/24/98 09:26p 49,152 ORDPRT.DLL
03/24/98 07:46p 886,784 CSMSUBS.DLL
09/30/97 06:42p 46,320 MAGMCTL.DLL
03/16/98 01:10p 18,432 RTITOOLS.DLL
01/30/98 10:20a 60,416 EQMODEL.DLL
01/30/98 09:59a 52,736 SFSALHIS.DLL
01/30/98 10:11a 376,832 PFIX.DLL
```
01/30/98 10:19a 63,488 CUSTALIS.DLL
03/24/98 08:03p 195,584 TRUECFSD.DLL
03/24/98 08:09p 122,880 ALERT.EXE
03/24/98 08:02p 151,552 CFS.EXE
03/24/98 09:22p 437,760 CFSNOTES.EXE
01/30/98 11:02a 55,296 ENNCTIME.DLL
01/30/98 09:59a 40,960 SFLTSENT.DLL
03/13/98 10:22p 34,304 CONVERT.EXE
03/24/98 08:04p 128,512 CTLADM.EXE
01/30/98 10:58a 131,072 WORKITEM.DLL
09/24/97 08:09a 154,624 HTMLGEN.EXE
11/17/97 11:52a 329 CFS1.LNK
03/24/98 08:14p 135,680 CTLCFS.EXE
03/24/98 08:06p 125,440 CTLRPT.EXE
03/18/98 07:37p 97,792 MNTSERCH.DLL
03/24/98 03:52p 34,816 CFSSCI.DLL
06/09/96 03:40p 22,478 CTLFIR.BMP
01/30/98 10:31a 82,944 CHARVSET.DLL
03/17/98 11:17a 52,736 CONSRCR.DLL
03/24/98 08:11p 128,512 ESSERVER.EXE
03/24/98 08:01p 124,928 HTMLGEN.EXE
03/24/98 09:15p 152,576 RTIUPG.EXE
03/24/98 08:07p 154,624 OPS.EXE
09/27/96 11:22a 580 LCABOX.EXP
03/24/98 08:06p 125,440 RPT.EXE
12/19/96 09:53p 2,054 RTILOAD.LIB
01/30/98 10:41a 55,296 SLCUST.DLL
01/30/98 10:32a 127,488 CHARMOD.DLL
01/30/98 10:38a 82,944 CHARTSET.DLL
03/24/98 10:33a 145,408 CHAVSET.DLL
03/17/98 11:17a 52,736 CONSRCR.DLL
07/09/97 01:48a 49,684 TSKSRES.DLL
01/30/98 10:22a 74,752 CONTEQU.DLL
01/30/98 10:46a 19,968 R0003.DLL
09/23/94 06:07p 115,462 NEWCFS.BMP
01/30/98 11:15a 39,936 RTILOAD.BPL
03/17/98 11:16a 167,936 RTILOAD.RUL
01/30/98 10:57a 102,400 CPERNOTE.DLL
01/30/98 11:07a 59,904 MCACTIVE.DLL
03/24/98 09:26a 12,288 CPUNCHAR.DLL
03/24/98 09:03p 119,296 RTILOAD.EXE
03/24/98 08:05p 186,880 RTILOAD.DLL
03/24/98 08:09p 152,064 CTLALRTD.DLL
03/24/98 08:03p 207,872 CTLCFSD.DLL
01/30/98 11:09a 53,760 MSCHRGRN.DLL
04/16/97 10:59a 1,988 ESSERVER.BAK
03/23/98 08:09p 122,880 TRUALRT.EXE
03/24/98 08:14p 145,920 SLS.EXE
03/24/98 08:14p 145,920 SLS.EXE
07/04/97 11:26a 254,857 CFSUSER.HLP
03/04/98 03:09p 329 CFS2.LNK
03/24/98 08:03p 122,880 TRUECFSD.DLL
07/09/97 01:57a 46,592 PFSRCRES.DLL
03/10/98 10:25a 75,264 LICALL.DLL
01/30/98 10:34a 78,848 CFSNOTES.DLL
03/24/98 08:06p 159,744 TRUECFSD.DLL
01/30/98 10:17a 47,616 DEBUG.DLL
03/24/98 06:09p 115,712 TRUECFSD.DLL
07/09/97 01:47a 56,320 CSRCRES.DLL
Oracle 8 File List

The following is a complete list of server exe & dll's for Oracle 8 on a WindowsNT system.
06/14/97 03:43p 14,336 NMP80.DLL
06/14/97 03:43p 10,752 NMS80.DLL
06/14/97 03:43p 45,056 NNCI80.DLL
06/14/97 03:43p 5,120 NNFD80.DLL
06/14/97 03:43p 29,696 NNFG80.DLL
06/14/97 03:43p 16,384 NNFN80.DLL
06/14/97 03:43p 121,344 NNG80.DLL
06/14/97 03:43p 17,408 NPL80.DLL
06/14/97 03:43p 119,296 NR80.DLL
06/14/97 03:41p 294,400 NS80.DLL
06/17/97 09:09p 27,648 NT80.DLL
06/18/97 03:41p 25,088 NTN80.DLL
06/18/97 03:41p 17,920 NTP80.DLL
06/14/97 03:44p 34,816 NTS80.DLL
06/17/97 09:08p 53,248 NTT80.DLL
06/18/97 03:41p 25,088 NTUS80.DLL
06/14/97 03:43p 346,624 NZ80.DLL
06/17/97 11:03p 6,656 O80DBNOP.DLL
06/17/97 11:03p 6,656 O80DISOP.DLL
06/17/97 11:03p 6,656 O80HETOP.DLL
06/17/97 11:03p 6,656 O80OBJOP.DLL
06/17/97 11:03p 6,656 O80PAROP.DLL
06/17/97 11:03p 6,656 O80REPOP.DLL
06/08/97 06:10a 6,656 O80SPAOP.DLL
06/20/97 06:45p 6,656 O80VSNOP.DLL
06/16/97 02:14p 41,984 OCI803JDBC.DLL
06/20/97 01:30p 17,920 OCIW32.DLL
06/18/97 03:29a 17,408 OCOPY80.EXE
06/20/97 09:10p 92,160 OIBKND32.DLL
06/11/97 05:24p 211,968 OICORNLS.DLL
06/12/97 12:31p 284,160 OIEXEC.DLL
06/11/97 05:30p 85,504 OICORNLS.DLL
06/20/97 11:30a 76,800 OIEXEC.SUS
06/12/97 12:31p 284,160 OIIMPL32.DLL
06/20/97 01:36p 27,136 OIIMPL.DLL
06/11/97 05:30p 14,848 OIIMRSUS.DLL
05/09/97 03:38p 290,816 OIP21.DLL
05/07/97 04:21p 24,981 OIP21.TLB
05/21/97 05:08a 21,504 OISBRSUS.DLL
05/22/97 07:59a 13,824 OISFP10.EXE
05/21/97 05:09a 124,928 OISIFAC.DLL
05/21/97 05:07a 17,408 OISIFC.DLL
05/22/97 07:59a 93,696 OISIFCM.DLL
05/22/97 07:52a 22,016 OISLRUS.DLL
05/21/97 05:06a 44,544 OISOIFC.DLL
05/22/97 07:59a 151,040 OISOMP.DLL
05/22/97 07:59a 34,816 OISOMIMP.DLL
05/22/97 07:59a 86,528 OISSAC.DLL
05/22/97 07:50a 115,200 OISWIZ.DLL
06/19/97 12:12p 151,040 OIUIIMFC.DLL
06/13/97 12:23p 13,824 OMRESUS.DLL
06/14/97 11:42p 141,312 ONRSD80.EXE
06/16/97 01:10p 24,576 OPERF80.DLL
06/20/97 12:49p 1,646,592 ORA803.DLL
11/16/95 09:53a 148,992 ORAANSI.DLL
06/20/97 06:44p 8,371,200 ORACLE80.EXE
06/18/97 02:46p 38,400 SQLPLUS.EXE
06/05/97 02:29p 5,632 STRTDB80.EXE
06/18/97 03:00a 128,000 SVRMGR30.EXE
06/13/96 05:30p 187,392 TCL73.DLL
03/25/98 12:44p 6,144 testexpdat.txt
03/25/98 01:10p 6,144 testexpdata.dmp
06/18/97 03:28a 52,736 TKPROF80.EXE
06/17/97 10:16a 124,928 TNSLSNR80.EXE
06/14/97 03:44p 16,384 TNSPING80.EXE
06/14/97 03:47p 86,016 TRCASST.EXE
06/13/97 03:28p 1,460 TRCFMT.CMD
06/14/97 01:25a 477,184 UIRESUS.DLL
10/07/96 10:32p 92,160 UNZIP.EXE
06/17/97 03:26a 107,008 VAC.EXE
06/17/97 03:22a 20,480 VACRUS.DLL
06/17/97 03:47a 58,368 VAD.EXE
06/17/97 03:44a 16,896 VADRUS.DLL
06/17/97 12:11a 8,192 VAERUS.DLL
06/17/97 03:33a 87,040 VAG.EXE
06/17/97 03:30a 22,016 VAGRUS.DLL
06/17/97 03:36a 78,848 VALEXE
06/17/97 03:33a 21,504 VAIRUS.DLL
06/17/97 12:13a 67,584 VAO.DLL
06/17/97 12:39a 39,936 VAOB.DLL
06/17/97 12:38a 7,680 VAOBRUS.DLL
06/17/97 12:37a 215,040 VAOC.DLL
06/17/97 12:32a 22,016 VAOCRUS.DLL
06/17/97 03:44a 374,272 VAOD.DLL
06/17/97 03:37a 219,648 VAODRUS.DLL
06/17/97 12:41a 117,248 VAOG.DLL
06/17/97 12:39a 12,288 VAOGPRUS.DLL
06/17/97 01:08a 303,104 VAOL.DLL
06/17/97 01:01a 103,936 VAOIRUS.DLL
06/17/97 01:00a 900,608 VAOS.DLL
06/17/97 12:41a 204,800 VAOSRUS.DLL
06/17/97 03:22a 59,904 VAP.DLL
06/17/97 03:20a 22,016 VAPRUS.DLL
06/17/97 04:31a 872,960 VAR.EXE
06/17/97 03:51a 799,232 VARRUS.DLL
06/17/97 03:30a 160,768 VAS.DLL
06/17/97 03:27a 18,432 VASRUS.DLL
06/17/97 03:51a 92,672 VAW.EXE
06/17/97 03:47a 31,744 VAWRUS.DLL
06/17/97 12:31a 173,056 VAXA.DLL
06/17/97 12:28a 30,720 VAXARUS.DLL
06/17/97 12:00a 126,976 VAXC.DLL
06/17/97 12:13a 24,064 VAXCRUS.DLL
06/17/97 12:16a 211,456 VAXCT.DLL
06/16/97 11:59p 65,536 VAXN.DLL
06/17/97 12:17a 53,248 VAXS.DLL
06/17/97 01:51a 67,072 VAXT.DLL
06/17/97 01:50a 7,680 VAXTRUS.DLL
06/16/97 11:58p 28,672 VAXX.DLL
01/31/97 04:18p 469,504 VIEW32W.DLL
06/17/97 05:33a 60,416 VMEM.EXE
06/17/97 01:11a 62,464 VMEP.EXE
06/17/97 01:09a 29,696 VMEPC.DLL
06/19/97 05:05p 83,968 VOBMGR.DLL
06/20/97 02:22p 10,240 VOBSH.EXE
06/17/97 02:16a 996,864 VOC.EXE
06/17/97 01:12a 313,856 VOCRUS.DLL
06/19/97 03:53p 16,849 VOCUS.CNT
06/17/97 07:05p 508,065 VOCUS.HLP
06/17/97 01:49a 292,352 VOD.EXE
06/17/97 05:31a 178,176 VODA.EXE
06/17/97 05:32a 14,848 VODARUS.DLL
06/17/97 01:20a 433,152 VOJT.OCX
06/17/97 01:12a 45,568 VOJTRUS.DLL
06/19/97 05:37a 103,424 VOM.EXE
06/17/97 05:35a 80,384 VOMRUS.DLL
06/17/97 01:52a 110,592 VOT.OCX
06/17/97 12:28a 39,424 VOX.DLL
06/17/97 05:42a 65,024 VOXD.DLL
06/19/97 09:56a 225,280 VSBC80.EXE
06/19/97 10:45a 223,232 VSRCV80.EXE
01/31/97 04:18p 90,112 W001F32W.DLL
01/31/97 04:18p 62,464 W003F32W.DLL
01/31/97 04:18p 97,792 W4W909F.DLL
06/17/97 05:22p 388,608 WRAP80.EXE
06/03/97 03:48p 38,400 WRQUICK.EXE
06/17/97 03:18p 64,512 XA80.DLL
06/19/97 05:06p 40,448 XPCRT.DLL
06/19/97 05:03p 1,086,304 XPFC.DLL
06/19/97 05:01p 859,796 XPFCUS.DLL
06/19/97 05:04p 57,344 XPGUI.DLL
06/19/97 05:01p 38,400 XPGUIUS.DLL
06/19/97 05:04p 394,240 XPLBSQO.DLL
10/07/96 10:32p 91,136 ZIP.EXE
259 File(s) 82,332,701 bytes
2,308,807,680 bytes free

CustomerFirst Management Object Format (MOF)

// **************************************************
// Product: Customer First
// **************************************************
instance of CIM_Product as $prod001
{
    Vendor = "CustomerFirst";
    Name = "CustomerFirst";
    ProductNumber = "PROD-001";
    IdentifyingNumber = "TYR-99-003";
    Caption = "CustomerFirst Support Product";
    Description = "The product for everyone ";
    Version = "01.01.00";
};

// **************************************************
// The product in decomposed into three software features.
// **********************************************************
// Software Feature 001 : Incident Manager
//
instance of CIM_SoftwareFeature as $feat001
{
    Vendor = "First Always Co." ;
    ProductName = "CustomerFirst" ;
    ProductNumber = "PROD-001" ;
    IdentificationNumber = "TYR-99-003" ;
    Version = "01.01.00" ;
    Name = "Incident Management" ;
    Caption = " Your Favorite Incident Manager" ;
    Description = "The best." ;
};

// **********************************************
instance of CIM_ProductSoftwareFeatures // Link Incident Manager with Product
{
    Product = $prod001 ;
    Component = $feat001 ;
};

// Software Feature 002 : Problem Tracker
//
instance of CIM_SoftwareFeature as $feat002
{
    Vendor = "First Always Co." ;
    ProductName = "CustomerFirst" ;
    ProductNumber = "PROD-001" ;
    IdentificationNumber = "TYR-99-003" ;
    Version = "01.01.00" ;
    Name = "Problem Tracker" ;
    Caption = " Your Favorite Problem Manager" ;
    Description = "The best." ;
};

// **********************************************
instance of CIM_ProductSoftwareFeatures // Link Problem Tracking with Product
{
    Product = $prod001 ;
    Component = $feat002 ;
};

// Software Feature 003 : Advance Graphics Package
//
instance of CIM_SoftwareFeature as $feat003
{
    Vendor = "First Always Co." ;
    ProductName = "CustomerFirst" ;

ProductNumber = "PROD-001";
IdentificationNumber = "TYR-99-003";
Version = "01.01.00";
Name = "Advance Graphics Package";
Caption = "Cool Graphics Package";
Description = "The best.";
};
// **********************************************
instance of CIM_ProductSoftwareFeatures // Link Advance Graphics with Product
{
Product = $prod001;
Component = $feat003;
}

// **********************************************
// Each Software Feature is decomposed into a set of software elements.
// **********************************************

// The software elements for the Incident Management Features are:
// Software Element 1 (Depl/NT) for Incident Management Feature
// instance of CIM_SoftwareElement as $elmt001_01
{
Name = "Incident Manager Element"; // key
TargetOperatingSystem = 17; // NT
Version = "01.01.01 "; // key
SoftwareElementState = 0; // Deployable
SoftwareElementID = " Definition"; // key
Version = "01.01.01 "; // key
Manufacturer = "First Always, Co.";
SerialNumber = " ";
LanguageEdition = "en";
}

instance of CIM_SoftwareFeatureSoftwareElements
{
GroupComponent = $feat001;
PartComponent = $elmt001_01;
}

// Software Element 2 (Inst/NT) for Incident Management Feature
// instance of CIM_SoftwareElement as $elmt001_02
{
Name = "Incident Manager Element"; // key
TargetOperatingSystem = 17; // NT
Version = "01.01.01 "; // key
SoftwareElementState = 1; // Installable
SoftwareElementID = " Definition"; // key
Version = "01.01.01 "; // key
Manufacturer = "First Always, Co.";
SerialNumber = " ";
LanguageEdition = "en";
instance of CIM_SoftwareFeatureSoftwareElements
{
  GroupComponent = $feat001 ;
  PartComponent = $elmt001_02 ;
};

// Software Element 3 (Exec/NT) for Incident Management Feature
// instance of CIM_SoftwareElement as $elmt001_03
{
  Name = "Incident Manager Element"; // key
  TargetOperatingSystem = 17; // NT
  Version = "01.01.01 "; // key
  SoftwareElementState = 2 ; // Executable
  SoftwareElementID = " Definition"; // key
  Version = "01.01.01 "; // key
  Manufacturer = "First Always, Co. ";
  SerialNumber = " ";
  LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
  GroupComponent = $feat001 ;
  PartComponent = $elmt001_03 ;
};

// Software Element 4 (Run/NT) for Incident Management Feature
// instance of CIM_SoftwareElement as $elmt001_04
{
  Name = "Incident Manager Element"; // key
  TargetOperatingSystem = 17; // NT
  Version = "01.01.01 "; // key
  SoftwareElementState = 3 ; // Running
  SoftwareElementID = " Definition"; // key
  Version = "01.01.01 "; // key
  Manufacturer = "First Always, Co. ";
  SerialNumber = " ";
  LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
  GroupComponent = $feat001 ;
  PartComponent = $elmt001_04 ;
};

// Software Element 5 (Depl/WIN95) for Incident Management Feature
// instance of CIM_SoftwareElement as $elmt001_05
{
  Name = "Incident Manager Element"; // key
TargetOperatingSystem = 15; // WIN95
Version = "01.01.01 "; // key
SoftwareElementState = 0 ; // Deployable
SoftwareElementID = " Definition"; // key
Version = "01.01.01 "; // key
Manufacturer = "First Always, Co. ";
SerialNumber = " ";
LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
    GroupComponent = $feat001 ;
    PartComponent = $elmt001_05 ;
};

//
// Software Element 6 (Inst/WIN95) for Incident Management Feature
//
instance of CIM_SoftwareElement as $elmt001_06
{
    Name = "Incident Manager Element"; // key
    TargetOperatingSystem = 15; // WIN95
    Version = "01.01.01 "; // key
    SoftwareElementState = 1 ; // Installable
    SoftwareElementID = " Definition"; // key
    Version = "01.01.01 "; // key
    Manufacturer = "First Always, Co. ";
    SerialNumber = " ";
    LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
    GroupComponent = $feat001 ;
    PartComponent = $elmt001_06 ;
};

//
// Software Element 7 (Exec/WIN95) for Incident Management Feature
//
instance of CIM_SoftwareElement as $elmt001_07
{
    Name = "Incident Manager Element"; // key
    TargetOperatingSystem = 15; // WIN95
    Version = "01.01.01 "; // key
    SoftwareElementState = 2 ; // Executable
    SoftwareElementID = " Definition"; // key
    Version = "01.01.01 "; // key
    Manufacturer = "First Always, Co. ";
    SerialNumber = " ";
    LanguageEdition = "en";
};

instance of CIM_SoftwareFeatureSoftwareElements
{
Software Element 8 (Run/WIN95 for Incident Management Feature)

instance of CIM_SoftwareElement as $elmt001_08
{
Name = "Incident Manager Element"; // key
TargetOperatingSystem = 15; // WIN95
Version = "01.01.01"; // key
SoftwareElementState = 3; // Running
SoftwareElementID = "Definition"; // key
Version = "01.01.01"; // key
Manufacturer = "First Always, Co.";
SerialNumber = "";
LanguageEdition = "en";
};

// Each of the software elements have a series of checks defined for them.
// These are organized into the in-state checks and the next-state checkes.

// Feature Platform/State Phase
// _____________|_______________|_______________
// Incident | NT |
// Manager | | In-State
// Element | Deployable |

// Feature Platform/State Phase
// _____________|_______________|_______________
// Incident | NT |
// Manager | | Next-State
// Element | Deployable |

// Feature Platform/State Phase
// _____________|_______________|_______________
// Incident | NT |
// Manager | | In-State
// Element | Installable |
instance of CIM_OSVersionCheck as $chck001_03_N01
{
    Name = "Incident Manager Element"; // key
    TargetOperatingSystem = 17; // NT
    Version = "01.01.01"; // key
    SoftwareElementState = 1; // Installable
    SoftwareElementID = "Definition"; // key
    Version = "01.01.01"; // key
    CheckID = "OS VersionCheck";
    Description = "";
    Caption = "";
    CheckMode = True;
    MinimumVersion = "3.5.1";
    MaximumVersion = "3.5.1";
};
Instance of CIM_SoftwareElementChecks
{
    Element = $elmt001_03
    Check = $chck001_03_N01
    Phase = 1; // next-state
};

instance of CIM_OSVersionCheck as $chck001_03_I01
{
    Name = "Incident Manager Element"; // key
    TargetOperatingSystem = 17; // NT
    Version = "01.01.01"; // key
    SoftwareElementState = 2; // Executable
    SoftwareElementID = "Definition"; // key
    Version = "01.01.01"; // key
    CheckID = "OS VersionCheck";
    Description = "";
    Caption = "";
    CheckMode = True;
    MinimumVersion = "3.5.1";
    MaximumVersion = "3.5.1";
Instance of CIM_SoftwareElementChecks
{
  Element = $elm001_01 ;
  Check = $chck001_03_I01 ;
  Phase = 0 ; // in-state
};

instance of CIM_DiskSpaceCheck as $chck001_03_I02
{
  Name = "Incident Manager Element"; // key
  TargetOperatingSystem = 17; // NT
  Version = "01.01.01 "; // key
  SoftwareElementState = 2 ; // Executable
  SoftwareElementID = " Definition"; // key
  Version = "01.01.01 "; // key
  CheckID = "OS VersionCheck";
  Description = " ";
  Caption = " ";
  CheckMode = True;
  AvailableDiskSpace = 15000; // kilobytes
};

Instance of CIM_SoftwareElementChecks
{
  Element = $elm001_01 ;
  Check = $chck001_03_I02 ;
  Phase = 0 ; // in-state
};

instance of CIM_MemoryCheck as $chck001_03_I03
{
  Name = "Incident Manager Element"; // key
  TargetOperatingSystem = 17; // NT
  Version = "01.01.01 "; // key
  SoftwareElementState = 2 ; // Executable
  SoftwareElementID = " Definition"; // key
  Version = "01.01.01 "; // key
  CheckID = "OS VersionCheck";
  Description = " ";
  Caption = " ";
  CheckMode = True;
  MemorySize = 32000; // kilobytes
};

Instance of CIM_SoftwareElementChecks
{
  Element = $elm001_01 ;
  Check = $chck001_03_I03 ;
  Phase = 0 ; // in-state
};

instance of CIM_SoftwareElementVersionCheck as $chck001_03_I04
{
  Name = "Oracle"; // key
  TargetOperatingSystem = 17; // NT
  Version = "07.01.01 "; // key
  SoftwareElementState = 2 ; // Executable
  SoftwareElementID = " Definition"; // key
CheckID = "RelationalDatabaseCheck";
Description = " ";
Caption = " ";
CheckMode = True;
SoftwareElementName = "Problem Tracker";
LowerSoftwareElementVersion = "01.01.00";
UpperSoftwareElementVersion = "01.01.00";
SoftwareElementState = 2;
TargetOperatingSystem = 17;
};
Instance of CIM_SoftwareElementChecks
{
Element = $elmt001_01;
Check = $chck001_03_I04;
Phase = 0; // in-state
};

// **************************************
// Feature Platform/State Phase
// Incident | NT |
// Manager | | Next-State
// Element | Executable |
//

instance of CIM_MemoryCheck as $chck001_01_03
{
Name = "Incident Manager Element"; // key
TargetOperatingSystem = 17; // NT
Version = "01.01.01"; // key
SoftwareElementState = 2; // Executable
SoftwareElementID = " Definition"; // key
Version = "01.01.01"; // key
CheckID = "MemoryCheck";
Description = " ";
Caption = " ";
CheckMode = True;
MemorySize = 32000; // units ("KiloBytes")
};
instance of CIM_SoftwareElementChecks
{
Element = $elmt001_01;
Check = $chck001_01_03;
Phase = 1; // next-state
};

// **************************************
// Feature Platform/State Phase
// Incident | NT |
// Manager | | In-State
// Element | Running |

// Feature Platform/State Phase
// Incident | NT |
// Manager | Next-State
// Element | Running |
Application Management Object Format (MOF)

// Application System
// **********************************************
instance of CIM_ApplicationSystem
{
  CreationClassName = "CIM_ApplicationSystem";
  Name = "WeServiceIt Customer Support";
};

// Application System Software Feature Associations
// **********************************************
instance of CIM_ApplicationSystemSoftwareFeatures
{
  GroupComponent = "
  CIM_ApplicationSystem.CreationClassName = "CIM_ApplicationSystem" ,
  CIM_ApplicationSystem.Name = "WeServiceIt Customer Support"
  ",
  PartComponent = "
  CIM_Product.Vendor = "First Always Co." ,
  CIM_Product.ProductName = "CustomerFirst" ,
  CIM_Product.ProductNumber = "PROD-001" ,
  CIM_Product.IdentificationNumber = "TYR-99-003" ,
  CIM_Product.Version = "01.01.00" ,
  CIM_SoftwareFeature.Name = "Incident Management"
  ",
};
instance of CIM_ApplicationSystemSoftwareFeatures
{
  GroupComponent = "
  CIM_ApplicationSystem.CreationClassName = "CIM_ApplicationSystem" ,
  CIM_ApplicationSystem.Name = "WeServiceIt Customer Support"
  ",
  PartComponent = "
  CIM_Product.Vendor = "First Always Co." ,
  CIM_Product.ProductName = "CustomerFirst" ,
  CIM_Product.ProductNumber = "PROD-001" ,
  CIM_Product.IdentificationNumber = "TYR-99-003" ,
  CIM_Product.Version = "01.01.00" ,
  CIM_SoftwareFeature.Name = "Problem Management"
  ",
};
instance of CIM_ApplicationSystemSoftwareFeatures
{
  GroupComponent = "
  CIM_ApplicationSystem.CreationClassName = "CIM_ApplicationSystem" ,
  CIM_ApplicationSystem.Name = "WeServiceIt Customer Support"
  ",
  PartComponent = "
  CIM_Product.Vendor = "First Always Co." ,
  CIM_Product.ProductName = "CustomerFirst" ,
  CIM_Product.ProductNumber = "PROD-001" ,
  CIM_Product.IdentificationNumber = "TYR-99-003" ,
  CIM_Product.Version = "01.01.00" ,
  CIM_SoftwareFeature.Name = "Problem Management"
  ",
};
CIM_Product.ProductName = "CustomerFirst" ,
CIM_Product.ProductNumber = "PROD-001" ,
CIM_Product.IdentificationNumber = "TYR-99-003" ,
CIM_Product.Version = "01.01.00" ,
CIM_SoftwareFeature.Name = "Database Server"
";
};

instance of CIM_ApplicationSystemSoftwareFeatures
{
GroupComponent = "
CIM_ApplicationSystem.CreationClassName = "CIM_ApplicationSystem" ,
CIM_ApplicationSystem.Name = "WeServiceIt Customer Support"
";
PartComponent = "
CIM_Product.Vendor = "First Always Co." ,
CIM_Product.ProductName = "CustomerFirst" ,
CIM_Product.ProductNumber = "PROD-001" ,
CIM_Product.IdentificationNumber = "TYR-99-003" ,
CIM_Product.Version = "01.01.00" ,
CIM_SoftwareFeature.Name = "Advance Graphics Package"
";
};