



PROJET « Gestion de réseaux »

Projet : Web-Based Enterprise Management

Présenté par : BOU SABA Elie

Sous l'encadrement de : Mr.N. MBAREK

Table de matières

Liste des tableaux.....	2
Acronymes et abréviations	3
Remerciement.....	4
1. Introduction Générale :	5
2. Chapitre 1 : WBEM	5
2.1 Introduction :	5
2.2 WBEM :	5
2.2.1 Introduction sur WBEM :	5
2.2.2 Architecture du WBEM :	6
2.2.3 Description de l'architecture :	7
2.3 Comparaison entre WBEM et SNMP :	9
2.3.1) Gestion :	9
2.3.2) Performance :	9
2.3.3) Simplicité :	9
2.3.4) Sécurité :	10
2.3.5) Protocole :	10
2.4 Conclusion :	10
3. Chapitre 2 : Présentation de différents outils :	11
3.1 Introduction :	11
3.2 OpenPegasus :	11
3.3 WBEM services :	11
3.4 OpenWBEM :	12
3.5 PyWBEM :	12
3.6 Conclusion :	12
4. Chapitre 3 : Utilisation de PyWBEM :	13
4.1 Introduction :	13
4.2 Outil pywbemcli :	13
4.3 Quelques opérations avec pywbemcli :	13
4.4 Conclusion :	15

5. Conclusion Générale :	15
Références bibliographiques :	16
Annexe :	17

Liste de figures

Figure 1 : Architecture de WBEM	6
Figure 2 : Exemple d'un modèle CIM	7
Figure 3 : Exemple d'une classe de format MOF	8

Liste des tableaux

Tableau 1 : Acronymes et abréviations.....	3
--	---

Acronymes et abréviations

Tableau 1 : Acronymes et abréviations

API	Application programming interface
CIM	Common information model
DMTF	Distributed Management Task Force
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JCP	Java Community Process
JDBC	Java database connectivity
JNI	JAVA Native Interface
JSR	Java Specification Request
LDAP	Lightweight Directory Access Protocol
MIB	Management information base
MOF	Managed Object Format
NE	Network element
RMI	Remote Method Invocation
SNMP	Simple Network Management Protocol
WBEM	Web-Based Enterprise Management

Remerciement

Nos remerciements s'adressent à Monsieur MBAREK Nader de nous avoir introduit à ce sujet tout en nous apportant des explications tout au long du module qui nous ont permis de bien mener à ce projet et d'apprendre de nouvelles notions concernant la gestion de réseaux.

1. Introduction Générale :

Au cours des dernières années, le monde de l'informatique a vécu une évolution et une amélioration très rapide permettant aux différentes entreprises de travailler plus efficacement et d'une manière plus flexible avec leur divers équipements informatiques.

Avec cet avancement de la technologie, la gestion des ressources comme les ordinateurs, commutateurs ou autres implémentés dans l'entreprise est devenu plus complexe pour l'administrateur à maintenir. Une des solutions pour faire face à ce problème est d'utiliser le WBEM (Web-Based Enterprise Management) qui va être présenté en détails dans ce rapport.

Ce rapport est composé de trois parties principales. Le premier chapitre présentera le concept de la WBEM. Le deuxième chapitre indiquera les différents outils qui peuvent être utilisés pour le WBEM. On finira avec le troisième chapitre qui introduira quelques opérations faites en utilisant l'outil PyWBEM.

2. Chapitre 1 : WBEM

2.1 Introduction :

Dans ce premier chapitre, nous avons défini le standard WBEM qui peut être appliqué dans les domaines de la gestion en présentant l'architecture utilisée par ce standard et ses divers composants. Ensuite, une comparaison sur les différents aspects a été faite entre le standard WBEM et le standard SNMP.

2.2 WBEM :

2.2.1 Introduction sur WBEM :

L'initiative WBEM qui a été lancée en juillet 1996 par BMC Software, Cisco Systems, Compaq, Intel et Microsoft est maintenant largement adoptée. WBEM est basé sur les standards Internet et les standards ouverts DMTF (Distributed Management Task Force) [1]

WBEM est un ensemble de techniques et de standards Internet de gestion qui permet la gestion de systèmes hétérogènes en utilisant des équipements de fournisseurs différents. L'objectif principal de l'initiative WBEM est d'unifier la gestion des environnements d'informatique distribuée en une seule norme.

WBEM est utilisé de plus en plus par les fournisseurs d'équipements qui lancent des produits prenant en charge WBEM par exemple Microsoft et Sun ont prouvé leur engagement envers WBEM en permettant à Windows NT et Solaris de fonctionner en tant que serveurs WBEM. [2]

2.2.2 Architecture du WBEM :

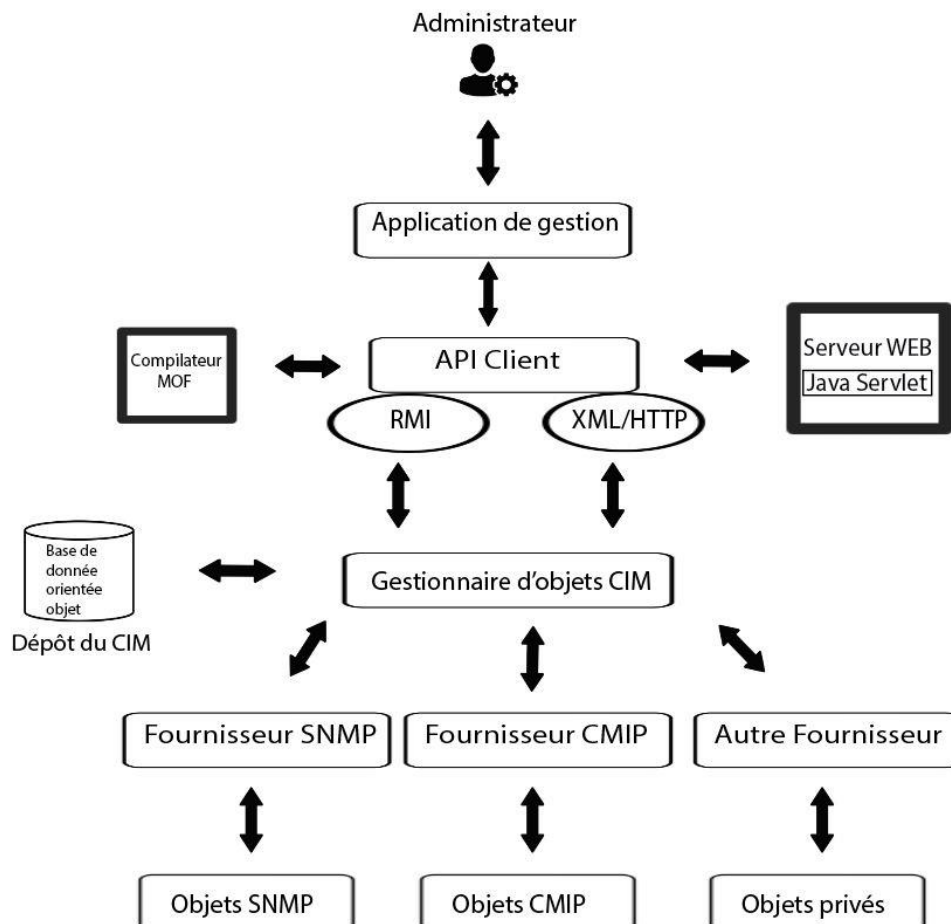


Figure 1 : Architecture de WBEM

2.2.3 Description de l'architecture :

L'architecture proposée par le DMTF assure une gestion homogène d'un ensemble hétérogènes de ressources. Pour construire un système compatible WBEM, on a besoin d'un client WBEM et au moins un serveur WBEM. L'Administrateur système est le client qui utilise une application de gestion pour gérer un objet SNMP par exemple qui est considéré comme le serveur WBEM dans ce cas. L'application de gestion adopte le client API (Application programming interface) pour transférer des données vers et depuis le gestionnaire d'objets, par exemple demander ou mettre à jour des informations sur un objet. CIM (Common information model) développé par le DMTF est une norme industrielle utilisée pour gérer les systèmes et les réseaux. C'est une représentation orientée objet comportant des modèles standard pour les différents équipements informatiques et qui a pour but de modéliser toutes sortes d'informations avec le même modèle d'information.

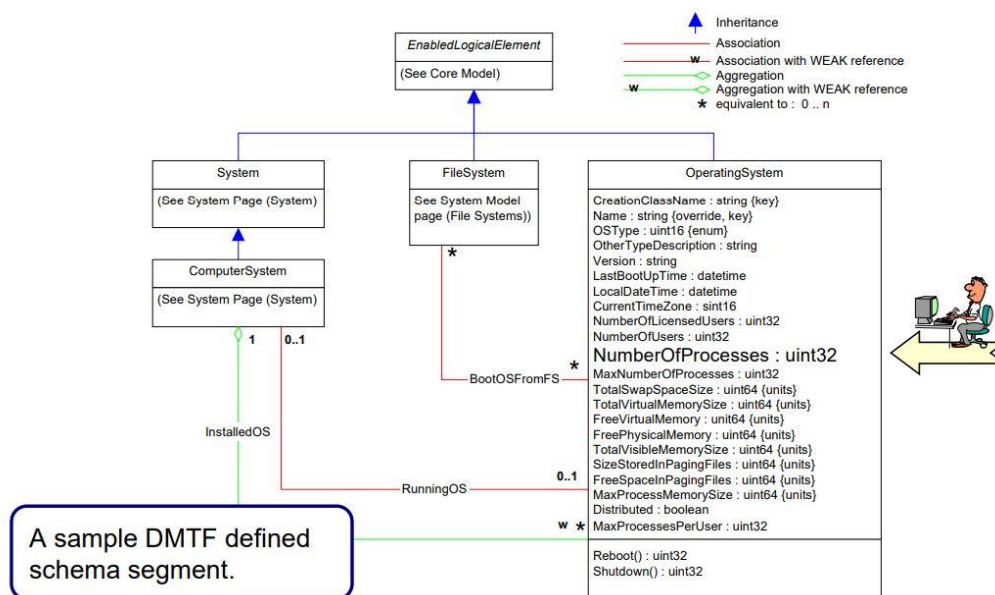
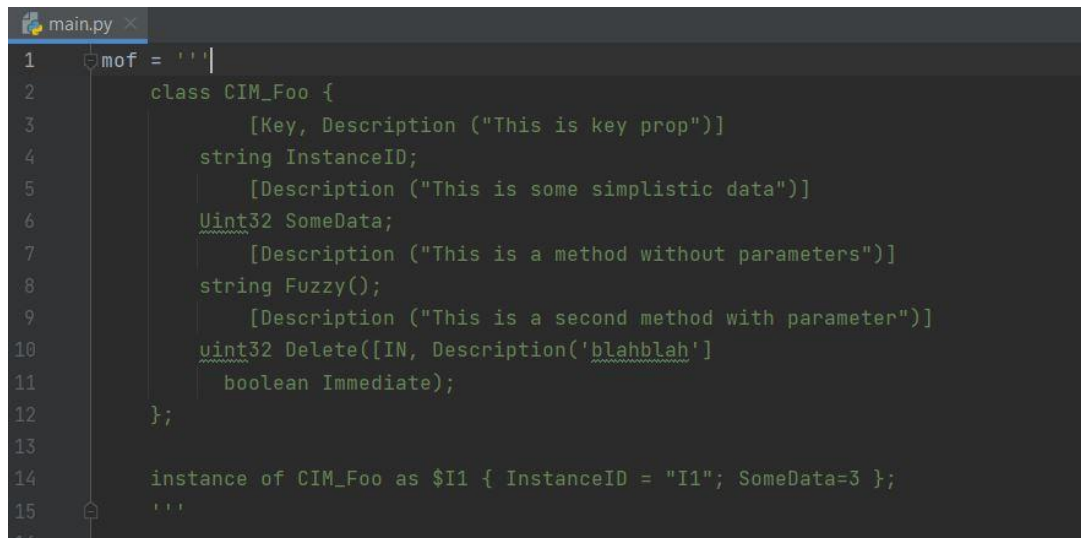


Figure 2 : Exemple d'un modèle CIM

L'API client communique avec le gestionnaire d'objets en se basant sur RMI (Remote Method Invocation) ou XML (Extensible Markup Language) via un protocole HTTP (HyperText Transfer Protocol) utilisant le serveur Web. Un objet est une représentation informatique ou un modèle d'une ressource gérée du système informatique. Le langage MOF (Managed Object Format) possède une syntaxe qui lui permet de définir des classes CIM. Le compilateur MOF va lire les fichiers créés dans le format MOF et par la suite les convertir en classe JAVA et ensuite

les stocker dans le dépôt CIM qui se base sur les espaces de nommages pour une organisation de manière hiérarchique et précis.



```
1 mof = ''
2 class CIM_Foo {
3     [Key, Description ("This is key prop")]
4     string InstanceID;
5     [Description ("This is some simplistic data")]
6     Uint32 SomeData;
7     [Description ("This is a method without parameters")]
8     string Fuzzy();
9     [Description ("This is a second method with parameter")]
10    uint32 Delete([IN, Description('blahblah')
11        boolean Immediate);
12 };
13
14 instance of CIM_Foo as $I1 { InstanceID = "I1"; SomeData=3 };
15 ''
16
```

Figure 3 : Exemple d'une classe de format MOF

Le logiciel serveur web peut être n'importe quel WBEM approprié, par exemple serveur Web compatible XML tel que Sun Web Server de Sun Microsystems, Inc. Le JAVA servlet va convertir les données XML et les réachemine au client API en format RMI. Cependant, si le gestionnaire d'objets est compatible avec le format HTTP, l'API client peut communiquer directement avec le gestionnaire d'objets sans avoir besoin du serveur web pour convertir les données. L'API client établit une connexion réseau (connexion local ou distante) pour accéder au gestionnaire d'objets CIM. Le dépôt API est utilisé par le gestionnaire d'objets pour stocker ou extraire des données avec le dépôt du CIM qui est une zone de stockage centrale pour la classe CIM. Le dépôt API est composé par une variété de classes, d'instances et de méthodes nécessaire pour communiquer avec le dépôt CIM en utilisant le protocole adapté. La connexion entre eux s'effectue en utilisant le protocole JDBC (Java database Connectivity) pour faire accès à la base de données orientée objet.

Lorsque le gestionnaire d'objets reçoit une requête de l'application de gestion pour demander ou mettre à jour des informations sur un objet géré, il recherche dans le dépôt CIM et si les données ne sont pas disponibles dans le dépôt CIM, il transmet la demande à un fournisseur d'objet qui est implémenté sur la même machine que le gestionnaire d'objets. Ensuite, le fournisseur d'objet utilise le fournisseur API pour communiquer avec les fournisseurs locaux qui sont des classes qui exécutent diverses fonctions en réponse à une demande du gestionnaire d'objets. [3]

2.3 Comparaison entre WBEM et SNMP :

Les utilisateurs confondent souvent WBEM et SNMP car leurs fonctionnalités semblent similaires. Mais en fait, il existe des différences sur plusieurs aspects entre ces 2 standards ce qui induit à l'avantage d'utiliser un standard par rapport à l'autre et cela dépend de ce que l'utilisateur souhaite mettre en place avec le coût correspondant.

2.3.1) Gestion :

Alors que SNMP utilise des variables pour la gestion, WBEM s'appuie sur des principes orientés objet pour exposer et gérer les propriétés et donc utilise des instances d'objets correspondant à des classes définies dans des espaces de nommage précises en appliquant des relations entre les différentes classes. Cela rend la gestion plus simple et gérable pour un système volumineux et complexe. [4]

2.3.2) Performance :

Un client WBEM n'a besoin d'émettre qu'un seul appel en spécifiant le bon argument pour qu'il reçoit toutes les instances correspondantes à sa demande, tandis qu'en SNMP (Simple Network Management Protocol) le client doit avoir une connaissance préalable du modèle objet utilisé par le NE (network element) qui a l'accès à la MIB et cela est fait en effectuant plusieurs opérations 'get-next' qui va rendre l'attribut demandé et l'adresse du suivant. [5]

2.3.3) Simplicité :

SNMP a été conçu pour surveiller les périphériques réseau qui sont simples et nécessitent des solutions de gestion légères où SNMP fait du bon travail. Néanmoins, la technologie a bien avancé et comme conséquence à cette évolution une gestion plus complexe est demandée ce qui est au-delà de la simple nature de SNMP. Les équipements d'aujourd'hui sont plus puissants au niveau matériel et par la suite utilisent des systèmes d'exploitation plus avancés capables d'appliquer beaucoup plus des fonctionnalités et par la suite la gestion de ces équipements est plus complexe et nécessite des solutions de gestion plus puissantes là où SNMP ne s'intègre pas parfaitement. [4]

2.3.4) Sécurité :

SNMPv3 (Simple Network Management Protocol version 3) se base sur USM (User-based Security Model) qui assure l'authentification en utilisant les algorithmes sécurisés (HMAC-MD5-96 et HMAC-SHA-96) et le chiffrement des messages avec la méthode du chiffrement symétrique (CBC-DES).

SNMPv3 offre le VACM (View-based Access Control Model) qui est un mécanisme de contrôle d'accès spécifiant le droit d'accès des utilisateurs comme le droit d'accès à l'écriture et la lecture. Comparant avec WBEM, ce dernier n'offre pas un standard pour le contrôle d'accès mais cela dépend de celui qui met en œuvre la CIM pour modéliser l'accès aux différents utilisateurs.

En revanche, SNMP3 adopte UDP (User Datagram Protocol), différemment de WBEM qui se base sur HTTP et se réfère à TCP (Transmission Control Protocol). Par la suite, SNMP3v3 ne garantit pas la livraison des messages et il est plus susceptible aux attaques.[5]

2.3.5) Protocole :

WBEM est une initiative de plusieurs entreprises et utilise aujourd'hui le protocole HTTP pour l'échange des informations. Par la suite, l'utilisation de WBEM n'est pas rattachée à l'utilisation de leur propre protocole comme est le cas pour SNMP qui est un protocole et cela l'empêche de suivre l'évolution des exigences ce qui donne à WBEM une comptabilité robuste pour le futur si par exemple HTTP à été remplacé par un autre protocole.[4]

2.4 Conclusion :

Pour conclure, on remarque l'importance du WBEM avec l'évolution de la technologie ce qui demande un travail plus complexe au niveau de la gestion des équipements dans une entreprise et tout ça en répondant aux lacunes de SNMP. En plus, WBEM se place bien pour permettre la comptabilité avec des équipements de différents fabricants en adoptant le gestionnaire d'objets CIM ce qui lui donne aussi un vrai avantage par rapport aux autres standard.

3. Chapitre 2 : Présentation de différents outils :

3.1 Introduction :

Plusieurs outils ont été développés par plusieurs entreprises pour mettre en place l'utilisation du concept WBEM. Dans ce chapitre, nous avons présenté plusieurs outils qui peuvent être utilisés comme OpenWBEM, OpenPegasus, WBEM services et PyWBEM en indiquant les composants caractérisant chaque outil.

3.2 OpenPegasus :

C'est un serveur CIM open source pour les objets DMTF CIM, écrit en langage C++ et comprend le gestionnaire d'objets CIM, un groupe d'interfaces définies et implémente des opérations CIM sur les opérations HTTP et leurs encodages HTTP cimxml. OpenPegasus peut être implémenté sur linux ou Windows. Il est maintenu conforme aux spécifications DMTF CIM et WBEM avec les exceptions notées dans la documentation. OpenPegasus inclue les composants suivants :

- Serveur CIM
- Des fournisseurs CIM
- Compilateur MOF
- Interfaces de fournisseur afin que les fournisseurs puissent être créés dans plusieurs langues comme Java, C++

3.3 WBEM services :

C'est une implémentation du WBEM en open-source JAVA comportant des APIs et des outils pour la communication entre le client et le serveur. Les APIs se base sur JSR 48 (Java Specification Request) développé sous l'encadrement du JCP (Java Community Process). WBEM services contient les composants suivants :

- Des API Java préliminaires
- Gestionnaire d'objets CIM
- Dépôt CIM
- Compilateur MOF

3.4 OpenWBEM :

OpenWBEM est une application WBEM open source écrite en C ++, adaptée aux applications commerciales et non commerciales. C'est la base du développement d'un cadre de gestion qui supprime les barrières entre les différentes plateformes et garantit une véritable interopérabilité. Les administrateurs peuvent utiliser OpenWBEM pour la gestion des équipements par exemple la surveillance et la configuration des équipements dans une entreprise. OpenWBEM comporte le serveur CIM et autres composants nécessaires pour une gestion qui se base sur le concept du WBEM. OpenWBEM est compatible sur les plateformes suivantes : Linux, Solaris, HP-UX, AIX, Mac OS X et Novell Netware.

3.5 PyWBEM :

PyWBEM est une bibliothèque open source écrite dans le langage de programmation Python qui fournit aux administrateurs système un moyen simplifié pour accéder aux objets et aux opérations CIM dans les serveurs WBEM. PyWBEM inclue l'outil 'pywbemcli' qui est une ligne de commande utilisé par le client WBEM pour exécuter des opérations vers le serveur WBEM. PyWBEM permet la communication entre le client WBEM et le serveur en se basant sur XML/HTTP en offrant aussi la fonctionnalité d'Écouteur d'indication WBEM qui peut écouter vers des réponses transmises par le serveur WBEM. En plus, PyWBEM possède l'utilité 'PyWBEM_mock' qui permet la simulation d'un serveur WBEM qui n'est pas un serveur WBEM réel et appliquer des opérations vers ce serveur simulé. PyWBEM peut être mis en place sur les systèmes d'exploitation linux et Windows avec une version Python 2.7 ou plus.

3.6 Conclusion :

En bref, plusieurs outils open-source peuvent être implémentés pour le WBEM qui sont écrits par diverses langages de programmation mais qui amène tous à faire de la gestion via le web mais chacun de ces implémentations peut avoir des avantages sur l'autre et donc les open-source sont toujours sous-développement pour une meilleure optimisation et expérience.

4. Chapitre 3 : Utilisation de PyWBEM :

4.1 Introduction :

Dans ce chapitre nous avons présenté quelques opérations qui peuvent être faites en utilisant PyWBEM qui se base sur Python et utilise une ligne de commande pour exécuter les commandes.

4.2 Outil pywbemcli :

L'outil pywbemcli écrit en Python est utilisé pour réaliser des opérations vers un serveur WBEM en utilisant les normes CIM / WBEM définies par le DMTF. Cet outil fournit plusieurs fonctionnalités :

- Gérer ou demander des informations sur des données CIM (classes CIM, instances CIM...) du serveur WBEM par exemple.
- Exécuter des fichiers MOF définissant des objets CIM dans un serveur simulé (mock server)
- Gérer des fonctionnalités concernant un serveur WBEM (espace de nommage CIM, informations sur la marque et la version du serveur WBEM...)
- Demander des informations détaillées sur les interactions CIM-XML avec le serveur WBEM, y compris les statistiques de temps et les détails du flux de données.

4.3 Quelques opérations avec pywbemcli :

a) Création d'un serveur simulé (mock server) en réalisant une connexion vers un fichier MOF (Le contenu du fichier MOF est dans l'annexe du rapport) :

```
$ pywbemcli -m tests/unit/simple_assoc_mock_model.mof connection save assoc1
```

b) Afficher l'arborescence des classes :

```
$ pywbemcli -n assoc1 class tree
```

Resultat :

root

```
+-- TST_FamilyCollection
+-- TST_Lineage
+-- TST_MemberOfFamilyCollection
+-- TST_Person
    +-- TST_Personsub
```

c) Demande d'une classe du serveur avec la commande get :

```
$ pywbemcli -n assoc1 class get TST_Person
```

Resultat :

```
class TST_Person {
    [Key ( true ),
        Description ( "This is key prop" )]
    string name;
    string extraProperty = "defaultvalue";
};
```

d) Demande d'une instance précise :

```
$ pywbemcli -n assoc1 instance get 'root/cimv2:TST_Person.name="Sofi"'
```

Resultat :

```
instance of TST_Person {
    name = "Sofi";
};
```

e) Énumération des instances de la classe TST_Person en traversant toutes les associations possibles :

```
$ pywbemcli -n assoc1 -o table instance associators TST_Person.?
```

Resultat :

Pick Instance name to process

0: root/cimv2:TST_Person.name="Saara"

1: root/cimv2:TST_Person.name="Mike"

2: root/cimv2:TST_Person.name="Sofi"
3: root/cimv2:TST_Person.name="Gabi"
4: root/cimv2:TST_PersonSub.name="Gabisub"
5: root/cimv2:TST_PersonSub.name="Sofisub"
6: root/cimv2:TST_PersonSub.name="Mikesub"
7: root/cimv2:TST_PersonSub.name="Saarasub"

4.4 Conclusion :

En conclusion, on remarque bien l'avantage de l'utilisation de l'outil PyWBEM avec le service de ligne de commande qu'il offre et qui permet une manipulation simple des opérations vers le dépôt CIM et aussi la création d'un serveur simulé pour réaliser des tests.

5. Conclusion Générale :

Pour conclure, l'approche WBEM a eu un intérêt fort auprès des industriels et cela est dû à l'utilisation de la CIM qui contient les informations nécessaires pour gérer des équipements et être basé sur le web pour assurer la communication entre les différents composants qui est un composant principal de l'initiative et qui répond à l'évolution des besoins et l'aspiration des utilisateurs à faire des opérations via le web.

Références bibliographiques :

[1] (Todd, What is WBEM? <https://www.itprotoday.com/compute-engines/what-wbem>, Jun 30, 1998)

[2] (WBEM (Web-Based Enterprise Management), <http://www.wbem.fr>)

[3] (MANAGEMENT, <https://patentimages.storage.googleapis.com/64/0e/83/98d889137505c3/US6976262.pdf>, Jun 14, 1999)

[4] (Cole, SNMP vs. WBEM, <https://www.gwyncole.com/wmi/documents/SNMPvsWBEM.pdf>)

[5] (Sandlund, <http://www.diva-portal.org/smash/get/diva2:1022687/FULLTEXT01.pdf>)

[6] (ORACLE, Solaris WBEM Developer's Guide, https://docs.oracle.com/cd/E18752_01/html/817-0366/chap-ov-8.html)

[7] (pywbem, <https://pywbem.readthedocs.io/en/latest/intro.html>)

Annexe :

Le contenu du fichier mof utilisé comme serveur simulé dans le chapitre 3 :

```
class TST_Person{
    [Key, Description ("This is key prop")]
    string name;
    string extraProperty = "defaultvalue";
    [ValueMap {"1", "2"}, Values {"female", "male"}]
    uint16 gender;
    [ValueMap {"1", "2"}, Values {"books", "movies"}]
    uint16 likes[];
};

class TST_Personsub : TST_Person{
    string secondProperty = "empty";
    [PUnit ("byte * 10^3")]
    uint32 counter;
};

[Association, Description(" Lineage defines the relationship "
    "between parents and children.") ]

class TST_Lineage {
    [key] string InstanceID;
    TST_Person ref parent;
    TST_Person ref child;
};

[Association, Description(" Family gathers person to family.") ]

class TST_MemberOfFamilyCollection {
    [key] TST_Person ref family;
    [key] TST_Person ref member;
};

[ Description("Collection of Persons into a Family") ]

class TST_FamilyCollection {
```

```

    [Key, Description ("This is key prop and family name")]
    string name;
};

// Define instances of TST_Person

instance of TST_Person as $Mike {
    name = "Mike";
    likes = {1, 2};
    gender = 2;
};

instance of TST_Person as $Saara {
    name = "Saara";
    likes = {1};
    gender = 1;
};

instance of TST_Person as $Sofi {
    name = "Sofi";
    gender = 1;
};

instance of TST_Person as $Gabi{
    name = "Gabi";
    likes = {2};
    gender = 1;
};

// Define instances of the TST_PersonSub

instance of TST_PersonSub as $Mikesub{ name = "Mikesub";
    gender = 2;
    secondProperty = "one" ;
    counter = 1; };

instance of TST_PersonSub as $Saarasub { name = "Saarasub";
    gender = 1;

```

```

        secondProperty = "two" ;
        counter = 2; };

instance of TST_PersonSub as $Sofisub{ name = "Sofisub";
        gender = 1;
        secondProperty = "three" ;
        counter = 3; };

instance of TST_PersonSub as $Gabisub{ name = "Gabisub";
        gender = 1;
        secondProperty = "four" ;
        counter = 4; };

// Define instances of TST_Lineage

instance of TST_Lineage as $MikeSofi
{
    InstanceID = "MikeSofi";
    parent = $Mike;
    child = $Sofi;
};

instance of TST_Lineage as $MikeGabi
{
    InstanceID = "MikeGabi";
    parent = $Mike;
    child = $Gabi;
};

instance of TST_Lineage as $SaaraSofi
{
    InstanceID = "SaaraSofi";
    parent = $Saara;
    child = $Sofi;
};

```

```
};
```

```
// Define instances of TST_FamilyCollection
```

```
instance of TST_FamilyCollection as $Family1
```

```
{
```

```
    name = "family1";
```

```
};
```

```
instance of TST_FamilyCollection as $Family2
```

```
{
```

```
    name = "Family2";
```

```
};
```

```
instance of TST_MemberOfFamilyCollection as $SofiMember
```

```
{
```

```
    family = $Family1;
```

```
    member = $Sofi;
```

```
};
```

```
instance of TST_MemberOfFamilyCollection as $GabiMember
```

```
{
```

```
    family = $Family1;
```

```
    member = $Gabi;
```

```
};
```

```
instance of TST_MemberOfFamilyCollection as $MikeMember
```

```
{
```

```
    family = $Family2;
```

```
    member = $Mike;
```

```
};
```

```
[7]
```