Maintenance de stockage via Powershell

Une image contenant personne, technologie, intérieur, ordinateur portable

Description générée automatiquement

Figure - Image générée par Bing AI représentant le projet

Yann Mangiagli – FIN2

Vennes

88H

Gaël Sonney

Experts : Daniel Berney, Gabriel Maret

Table des matières

[1 Résumé de la documentation / résumé du rapport 3](#_Toc166826240)

[1.1 Comptes utilisés 3](#_Toc166826241)

[1.2 Résumé 3](#_Toc166826242)

[2 Spécifications 4](#_Toc166826243)

[2.1 Titre 4](#_Toc166826244)

[2.2 Description 4](#_Toc166826245)

[2.3 Matériel et logiciels à disposition 4](#_Toc166826246)

[2.4 Prérequis 4](#_Toc166826247)

[2.5 Cahier des charges 4](#_Toc166826248)

[2.5.1 Objectifs et portée du projet (objectifs SMART) 4](#_Toc166826249)

[2.5.2 Caractéristiques des utilisateurs et impacts 5](#_Toc166826250)

[2.5.3 Fonctionnalités requises (du point de vue de l’utilisateur) 5](#_Toc166826251)

[2.5.4 Contraintes 5](#_Toc166826252)

[2.5.5 Travail à réaliser par l'apprenti 5](#_Toc166826253)

[2.5.6 Si le temps le permet … 6](#_Toc166826254)

[2.5.7 Méthodes de validation des solutions 6](#_Toc166826255)

[2.6 Les points suivants seront évalués 6](#_Toc166826256)

[2.7 Validation et conditions de réussite 6](#_Toc166826257)

[3 Planification Initiale 6](#_Toc166826258)

[3.1.1 Mises à jour du planning 6](#_Toc166826259)

[4 Analyse 7](#_Toc166826260)

[4.1 Opportunités 7](#_Toc166826261)

[4.1.1 Approfondir 7](#_Toc166826262)

[4.1.2 Difficultés potentielles 7](#_Toc166826263)

[4.1.3 Matériel à exploiter 7](#_Toc166826264)

[4.1.4 Recherche d’informations particulière 7](#_Toc166826265)

[4.1.5 Solutions possibles 7](#_Toc166826266)

[4.2 Document d’analyse et conception 8](#_Toc166826267)

[4.3 Conception des tests 10](#_Toc166826268)

[4.4 Planification détaillée 10](#_Toc166826269)

[5 Réalisation 10](#_Toc166826270)

[5.1 Dossier de Réalisation 10](#_Toc166826271)

[5.2 Modifications 14](#_Toc166826272)

[6 Tests 14](#_Toc166826273)

[6.1 Dossier des tests 14](#_Toc166826274)

[7 Conclusion 15](#_Toc166826275)

[7.1 Bilan des fonctionnalités demandées 15](#_Toc166826276)

[7.2 Bilan de la planification 15](#_Toc166826277)

[7.3 Bilan personnel 15](#_Toc166826278)

[8 Divers 15](#_Toc166826279)

[8.1 Journal de travail 15](#_Toc166826280)

[8.2 Bibliographie 15](#_Toc166826281)

[8.3 Webographie 16](#_Toc166826282)

[9 Annexes 16](#_Toc166826283)

# Résumé de la documentation / résumé du rapport

## Comptes utilisés

|  |  |  |
| --- | --- | --- |
| **Utilisation** | **Compte** | **Mot de passe** |
| Envoi du mail automatisé | pappro2mail@gmail.com | .Etml-123 |
| Réception du mail automatisé | tpiymetml@gmail.com | .Etml-123 |
| Compte serveur YANN2K22SRV | Administrateur | .Etml-123 |
| Compte serveur YANN2K22SRV2 | Administrateur | .Etml-123 |

## Résumé

Le but du projet est de créer 2 scripts complémentaires. Le script 1 est un exécutant tandis que le script 2 sert à le propager automatiquement.

# Spécifications

## Titre

Maintenance automatique et régulière du stockage des postes de travail à distance

## Description

Les étudiants de la section informatique sont amenés à utiliser de nombreuses machines virtuelles pendant leur formation qu’ils et elles laissent sur leur lecteur (H :) qui leur servent de répertoire utilisateur sans les utiliser de nouveau. Ces machines virtuelles sont donc stockées en vain sur les machines de l’ETML plutôt que sur le SSD des étudiants et prennent beaucoup de place avec le temps. Le but de ce projet est de limiter l’impact des machines virtuelles sur le stockage des postes de l’école en avertissant quand le stockage d’un poste atteint 80% et en envoyant un mail au responsable qui pourra par la suite décider de supprimer les fichiers OVA de plus de 3Go des répertoires des élèves s’étant connecté à cette machine.

## Matériel et logiciels à disposition

PC standard de l’ETML sous Windows 10

PowerShell

PowerShell ISE

Deux machines distantes (WINSERV2022) pour le déploiement du script à distance avec les droits d’administrateur configurées par l’apprenti en amont.

## Prérequis

Module 431 - Planification de projets et de tâches

Module 122 – PowerShell

Module 117 – Réseau (Windows Server 2022, Câblage simple, Remote Desktop)

## Cahier des charges

### Objectifs et portée du projet (objectifs SMART)

Combien

Quoi

Qui

Comment

Où

Quand

Pourquoi

Spécifique

Mesurable

Atteignable

Réaliste : Tous les élèves doivent quand même pouvoir poser des fichiers volumineux sans que ceux-ci ne se fassent supprimer s’ils ne sont pas des ova.

Temporel

### Caractéristiques des utilisateurs et impacts

Les utilisateurs du script seraient des administrateurs système et réseau désirant optimiser le stockage des postes de travail supportant des machines virtuelles, comme le service informatique de l’ETML. Le but est donc d’automatiser les scripts tout en laissant aux administrateurs le moyen de personnaliser l’utilisation du script avec des paramètres. Par exemple, choisir la taille maximum tolérée par le script (par défaut 3Go comme mentionné plus haut).

### Fonctionnalités requises (du point de vue de l’utilisateur)

Pour le premier script, l’utilisateur doit pouvoir récupérer les informations concernant l’espace disque utilisé ainsi que l’espace libre, le nom de l’ordinateur, la version de son système, ainsi que les versions des mises à jour de l’OS. Le script envoie un courriel sur le mail d’entreprise de l’administrateur un avertissement quand 80% du stockage est utilisé, et doit pouvoir supprimer les fichiers trop volumineux, avec pourquoi pas un paramètre permettant de décider ce qui est considéré comme « trop volumineux ». Le deuxième script est plus simple et contient moins de fonctionnalités. L’utilisateur a juste besoin de le lancer pour qu’il copie le script 1 sur une machine à distance ne le possédant pas et qu’il crée la tâche planifiée correspondante. Le script peut gérer une liste de 10 ordinateurs en simultané pour son exécution.

### Contraintes

Plusieurs contraintes peuvent exister avec des scripts comme ceux-là. Par exemple, le mail envoyé par le script a une chance non négligeable de finir dans les spam, car il n’y a pas le temps d’installer un serveur SMTP et c’est pour cela qu’une adresse Gmail sera utilisée, car il est plus simple et plus rapide de se connecter via PowerShell sur ce serveur SMTP. Comme la sécurité n’est pas mentionnée dans le cahier des charges, la sécurité du script ne sera pas forcément optimale car cela ne sera pas une priorité. Par exemple, pour automatiser l’enregistrement des identifiants en utilisant la commande « get-credential », il faudra passer par écrire le mot de passe en clair dans le script pour ne pas perdre du temps là-dessus. Cela pourrait être un point à améliorer s’il reste du temps quand les 2 scripts seront prêts.

### Travail à réaliser par l'apprenti

L’apprenti doit réaliser 2 scripts ainsi qu’un rapport, une planification et un journal de travail. Le premier script doit récupérer des informations système comme le nom, version de l’OS, Espace disque libre & utilisé, version des mises à jour et envoyer un mailcontenant les informations permettant d’identifier le poste de travail à l’administrateur quand le stockage utilisé égale ou dépasse les 80%. Il sera ensuite possible de supprimer en priorité les fichiers de machine virtuelles, en particulier les fichiers avec une extension ova. Le deuxième script est une automatisation. Il permet de sélectionner une liste jusqu’à 10 ordinateurs distants, et d’implémenter le premier script afin qu’il puisse scanner et surveiller le stockage des postes de travail sur lesquels il vient de s’installer, tout en créant une tâche planifiée qui planifiera ce scan. Le rapport doit contenir toute la documentation qu’il a créée durant le projet ainsi que les tests réalisés. La planification

### Si le temps le permet …

Il serait judicieux d’améliorer la sécurité pour rendre le script utilisable par une entreprise ou une école, ainsi qu’installer un serveur SMTP pour ne pas avoir à utiliser une adresse gmail mais une adresse eduvaud dans l’exemple de l’ETML utilisé comme prétexte pour ce TPI.

### Méthodes de validation des solutions

Chaque fonctionnalité demandée dans le script sera testée en premier lieu seule pour être sûr que la solution fonctionne avant de l’incorporer dans le script 1 ou 2 suivant l’instruction(s) testée(s).

Comment les tests vont être entrepris, quels tests doivent être entrepris, etc.…

## Les points suivants seront évalués

* Le rapport
* Les planifications (initiale et détaillée)
* Le journal de travail
* Le code et les commentaires
* Les documentations de mise en œuvre et d’utilisation

## Validation et conditions de réussite

* Compréhension du travail
* Possibilité de transmettre le travail à une personne extérieure pour le terminer, le corriger ou le compléter
* Etat de fonctionnement du produit livré

# Planification Initiale

La technique des 6 pas est utilisée pour planifier le projet.

Le projet commence le 13 mai 2024 et se termine le lundi 3 juin 2024.

Le seul jour spécial est pentecôte, ce qui rend le lundi 20 mai 2024 chômé. Le lendemain, l’expert 2 vient voir l’avancée du projet.

### Mises à jour du planning

14.05.24 :

Modification de la rencontre avec expert 2, décidée arbitrairement dans la planification initiale par l’apprenti pour le 23 mai avant d’être modifié au 21 quand l’expert 2 et l’élève se sont mis d’accord.

Ce paragraphe présente le planning d'origine (date de début, date de fin, vacances et congés, liste hiérarchique des tâches ou GANTT, jalons, durée totale)

# Analyse

## Opportunités

### Approfondir

Il faudra approfondir certaines connaissances en PowerShell pour mener à bien le projet.

* Commande « New-ScheduledTaskAction » à apprendre à utiliser
* Apprendre à exécuter les scripts depuis un autre (Copy-Item ?)

### Difficultés potentielles

Plusieurs difficultés peuvent survenir, surtout à cause d’imprévus.

* Malgré le script d’envoi de mail fonctionnel réalisé en amont, rien ne promet qu’il fonctionnera dans un contexte différent
* Comment vérifier que le script 1 s’exécute bien à 80% alors que les machines à disposition ont des disques vides ?
* La copie du script sur une liste de machines n’a pas l’air simple à mettre en place
* Pour le script de copie, il faut copier un script sur la machine principale et le coller sur la machine distante
* Un serveur qui ne réagit pas comme l’autre

### Matériel à exploiter

Le matériel exploitable peut lui aussi causer des difficultés, par exemple un serveur mal paramétré en amont, ou encore si le réseau de l’école est changé. C’est pour cela qu’il est impossible d’utiliser le script depuis une machine distante sur la machine locale, car elle a besoin de droit administrateur que je ne possède pas, et elle est protégée derrière docker kubernetes.

### Recherche d’informations particulière

Comme PowerShell et ses erreurs sont bien documentées, il y’a peu de chances de rencontrer des difficultés de ce côté-ci. En revanche, un problème d’environnement rendrait tout de suite la tâche plus hardue.

### Solutions possibles

* Réduction du quota de 80% à 3% par exemple
* Une boucle « foreach » sur un tableau ayant la liste des ordinateurs
* Supprimer Invoke-Command, car on essayait d’accéder à la session depuis la session, c’est donc normal que ça ne fonctionnait pas
* Les pares-feux des 2 serveurs n’étaient pas identiques. Le premier était désactivé et pas l’autre.

Si les spécifications de départ ne laissent pas de doutes sur la manière de réaliser un projet, ce chapitre ne fera que renvoyer le lecteur aux spécifications.

## Document d’analyse et conception

La méthode de travail décidée à été la méthode des 6 pas.

### Script 1

Pour le script 1, des tests ayant été faits durant un projet précédent ont mis en exergue le fait qu’utiliser Send-MailMessage était obsolète et n’était plus supporté par Microsoft. Malgré le fait que des alternatives existent, il a été décidé de rester sur cette façon de faireé

### Script 2

Le deuxième script se focalise sur une façon d’utiliser le premier. En effet, celui-ci copie le premier script depuis la machine locale et le colle dans un répertoire spécifique sur la ou les machines distantes (jusqu’à un maximum de 10). C’est par celui-ci que le projet a été commencé, jugé plus simple à implémenter et tester.



Figure - Instanciation du tableau

Un tableau est créé comme paramètre afin de pouvoir mettre plusieurs adresses IP.



Figure - Instanciation d'un deuxième tableau

Un autre tableau d’une taille de 10 éléments est créé afin de pouvoir stocker les paramètres entrés dans $remoteip.



Figure - placeholder - boucle foreach

La boucle foreach permet de naviguer dans le tableau $remoteip et de récupérer toutes les adresses IP pour les ajouter dans $computerList mentionné ci-dessus. Out-Null est utilisé afin de masquer l’output fait par la boucle à chaque passage dans celle-ci.



Figure - Conversion du mot de passe

C’est dans cette partie du script que le mot de passe est choisi et stocké pour l’utiliser par la suite. Cette instruction cause des problèmes de sécurité car avoir un mot de passe en clair permet à n’importe quelle personne ayant accès au script de voir le mot de passe. Pour palier à cette problématique, une solution serait de créer un fichier avec les mots de passe hachés à l’intérieur, puis de venir le récupérer avec une commande spécifique.



Figure - Récupération du nom d'utilisateur

Le nom d’utilisateur est récupéré ici. Comme le mot de passe et le nom d’utilisateur sont identiques pour les 2 machines distantes, ils sont renseignés en dur dans le script. Si ce n’était pas le cas, des paramètres auraient été utilisés. Cela crée un problème de sécurité similaire à celui du mot de mot de passe mentionné précédemment.



Figure - Implantation des chemins

Pour ce qui est des chemins, ils sont aussi en dur. Pour régler ce problème il faudra créer une valeur avec le nom du fichier et faire une concaténation avec $PWD pour rendre le script plus indépendant de la machine utilisée. $startpath est le chemin à optimiser, pour que le script respecte entièrement le cahier des charges. C’est lui qui va chercher le script 1 pour le copier plus tard. $destinationPath est le chemin de destination où sera stocké le script copié plus tôt. La lettre C:\ a été choisie car c’est la lettre par défaut d’un stockage. On pourrait améliorer ça en permettant à l’administrateur de choisir le chemin de destination via un paramètre.

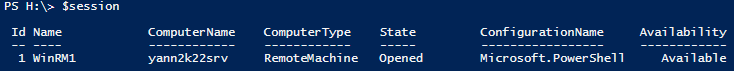


Figure - Début de la boucle

Cette boucle foreach commence par aller chercher le nom DNS des machines qui sera utile pour créer une session par la suite. C’est dans cette boucle qu’on utilise le tableau $computerList créé plus tôt.



Une session est créée ici pour chaque passage dans la boucle décidé par le nombre d’éléments dans le tableau $computerList cette session contient un id, un nom d’ordinateur, l’état de la connexion, le nom de la configuration qui s’apparente à comment la session a été configurée, et si la connexion et disponible ou non, comme le montre la capture d’écran ci-dessous.





Cette instruction copie le fichier situé au chemin de départ et se connecte à l’ordinateur distant grâce au l’instruction -ToSession, puis colle le script dans le chemin de destination situé dans l’ordinateur distant.



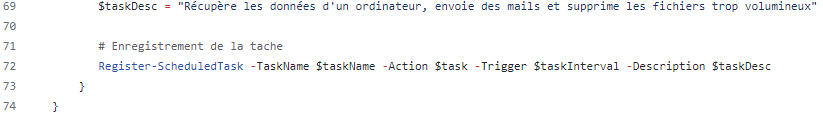
Création d’une variable utilisée pour avoir le chemin complet du script afin de pouvoir l’utiliser comme argument pour créer une tâche planifiée par la suite à l’intérieur de « Invoke-Command ». Celle-ci permet d’exécuter une commande sur la machine distante d’une manière plus efficace que Enter-PSSession, car Enter PSSession doit d’abord se connecter à la machine distante, tandis que invoke-command permet de rester sur la machine locale.



Une variable de tâche dénommée $task est créée en utilisant New-ScheduledTaskAction. On précise que la tâche utilisera PowerShell en le notant dans -Execute et on précise le script qui sera lancé grâce à son chemin.



$taskInterval consiste à décider de la fréquence à laquelle le script sera exécuté. Il s’exécutera donc tous les jours à 16h00. La suite du script donne un nom à la tâche et le stocke dans une variable du nom de $taskName.



Le script donne une description à la tâche, puis l’enregistre définitivement ensuite. Les 2 accolades permettent de refermer invoke-command ainsi que le foreach.

Ce paragraphe décrit le fonctionnement de manière détaillée.

Autant que possible de manière graphique, imagée, tableaux, etc.

Tous les cas particuliers devraient y être spécifiés…

Il s’agit d’y présenter les fonctionnalités à développer :

Découpage en étapes, en modules, en fonctionnalités, etc.

Formulaires, interfaces graphiques, pages web, etc.

Schémas de navigation, schémas événementiels, structogramme, pseudocode, etc.

## Conception des tests

La stratégie de test employée est triviale ; Chaque ligne est testée indépendamment des autres pour avoir une suite d’instructions qui fonctionnent. Une fois ces instructions mises ensembles, elles sont retestées par groupe jusqu’à qu’elles fonctionnent elles-aussi. Cette opération sera effectuée tant que le script ne fonctionne pas entièrement.

## Planification détaillée

A ce stade, après l’analyse complète du projet, un planning détaillé et complet (avec tâches, sous-tâches, dépendances, durée, …) peut être finalisé.

Le planning détaillé doit s’inscrire dans le planning initial. Il faut que l’on puisse situer cette planification détaillée par rapport à la planification initiale.

# Réalisation

## Dossier de Réalisation

### Version des outils logiciels utilisés

Machine locale

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.19041.4291 |

Yann2k22srv

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.20348.558 |

Yann2k22srv2

|  |  |
| --- | --- |
| **Logiciel** | **Version** |
| PowerShell Desktop | 5.1.20348.558 |

### Configuration spéciale des serveurs

Tout ce qui n’est pas précisé est un paramètre par défaut. Les 3 machines ont la police d’exécution déterminée sur « Unrestricted »

Les 2 machines distantes sont configurées de la même manière, la seule différence est que yann2k22srv2 a une autre adresse ip (192.168.10.53). Toutes les captures d’écran proviennent de yann2k22srv.

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

Figure - Spécifications de Windows Server

Une image contenant texte, capture d’écran, nombre, Police

Description générée automatiquement

Figure - Configuration du pare-feu

Les pares-feux sont désactivés car ils bloquent l’utilisation de la commande

$remotename = [System.Net.Dns]::GetHostByAddress($remoteip).Hostname

Cela empêche donc le script 2 de résoudre le DNS et de récupérer le nom de la machine qui est utilisé par la suite pour créer une session dans laquelle il sera possible de coller le script 1. En bloquant le pare-feu, on s’affranchit des problèmes du genre. Malgré tout, cela cause d’normes problèmes de sécurité. Le plus optimal serait de créer une règle qui autoriserait la commande de s’exécuter, et dans le même temps il faudrait désactiver toutes les règles qui coupent le script.

Une image contenant texte, capture d’écran, logiciel, Page web

Description générée automatiquement

Figure - Configuration de l'adresse IP

La partie réseau est décidée par le **réseau** de l’école et la partie hôte a été décidée arbitrairement. Comme c’est un masque /24, seulement la dernière partie est une partie hôte. Voici un tableau regroupant les adresses IP utilisées et à quelles machines elles correspondent :

|  |  |
| --- | --- |
| yann2k22srv | **192.168.10**.51 |
| INF-A13-M215 | **192.168.10**.52 |
| yann2k22srv2 | **192.168.10**.53 |

Une image contenant texte, logiciel, Police, affichage

Description générée automatiquement

Figure - Protocole IPV6 désactivé

Le protocole IPV6 est désactivé afin d’éviter les problèmes d’incompatibilité qui pourraient survenir en utilisant winRM

Cette partie permet de reproduire ou reprendre le projet par un tiers.

Pour chaque étape, il faut décrire sa mise en œuvre. Typiquement :

Versions des outils logiciels utilisés (OS, applications, pilotes, librairies, etc.)

Configurations spéciales des outils (Equipements, PC, machines, outillage, etc.)

Code source commenté des éléments logiciels développés.

Modèle physique d’une base de données.

Arborescences des documents produits.

Il faut décrire le parcours de réalisation et justifier les choix.

## Modifications

Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.

Date, raison, description, etc.

# Tests

## Dossier des tests

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Numéro | Description | Procédure | Résultat attendu | Résultat obtenu |
| 1 | Vérification synchronisation GitHub – GitHub desktop | Faire un commit GitHub en utilisant GitHub Desktop afin de voir si le commit passe bien sur GitHub | OK | OK |
| 2 | Lancement d’un script depuis son chemin à l’intérieur d’un autre script | Entrer la commande ./x-pappromail.ps1 pour voir si le script s’exécute | KO | OK |
| 3 | Affichage de l’aide (script 2) | Dans l’en tête des scripts, renseigner .SYNOPSIS, .DESCRIPTION, .EXAMPLE et .PARAMETER, puis faire les commandes  Get-Help x-yanmangiagl-setup.ps1 et  Get-Help yanmangiagl-setup.ps1 -Examples | L’aide s’affiche sans problème | L’aide s’affiche sans problème |
| 4 | Déplacement des variables (script 2) | Faire bouger les variables $startPath et $destinationPath dans le scriptblock d’invoke command pour ne plus avoir de x dite $null alors qu’elle ne l’est pas | Les variables sont instanciées et permettent au script de passer à la suite | Les 2 variables ont l’air de fonctionner maintenant mais $session est $null |
| 5 | Supprimer invokecommand |  | OK | OK |
| 6 | Tenter 192.168.10.53 sans l’autre comme paramètre du script 2 (.51 fonctionne) | Restart-Service winrm | Marche pour toujours | Fonctionne seulement une fois après un redémarrage du serveur si la commande est faite assez vite |
| 8 | Désactiver pare-feu serveur 2 | La premier serveur a un pare-feu éteint et pas le deuxième, tester de le désactiver pour voir si ça fonctionne | Marche pour toujours | Marche pour toujours |

On dresse le bilan des tests effectués (qui, quand, avec quelles données…) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats obtenus et les actions à entreprendre en conséquence (et une estimation de leur durée).

Si des tests prévus dans la stratégie n'ont pas pu être effectués :

raison, décisions, etc.

Liste des bugs répertoriés avec la date de découverte et leur état:

Corrigé, date de correction, corrigé par, etc.

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différence entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

Date, activité (description qui permet de reproduire le cheminement du projet), durée, liens et références sur des documents externes. Lorsqu’une activité de recherches a été entreprise, il convient d’énumérer ce qui a été trouvé, avec les références.

## Bibliographie

Références des livres, revues et publications utilisés durant le projet.

## Webographie

Références des sites Internet consultés durant le projet.

# Annexes

Listing du code source (partiel ou, plus rarement complet)

Guide(s) d’utilisation et/ou guide de l’administrateur

Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).

Extraits de catalogue, documentation de fabricant, etc.