Prerequisite Scanner
Version 1.2

Manuel d'utilisation

IBM

Prerequisite Scanner
Version 1.2

Manuel d'utilisation

IBM

Important
Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations générales figurant à la rubrique «Remarques», à la page 163.

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Table des matières

Figures

Tableaux

Avis aux lecteurs canadiens

Le présent document a été traduit en France. Voici les principales différences et particularités dont vous devez tenir compte.

Illustrations

Les illustrations sont fournies à titre d'exemple. Certaines peuvent contenir des données propres à la France.

Terminologie

La terminologie des titres IBM peut différer d'un pays à l'autre. Reportez-vous au tableau ci-dessous, au besoin.

IBM France	IBM Canada
ingénieur commercial	représentant
agence commerciale	succursale
ingénieur technico-commercial	informaticien
inspecteur	technicien du matériel

Claviers

Les lettres sont disposées différemment : le clavier français est de type AZERTY, et le clavier français-canadien de type QWERTY.

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

- les pages de codes 850 (multilingue) et 863 (français-canadien),
- le code pays 002,
- le code clavier CF.

Nomenclature

Les touches présentées dans le tableau d'équivalence suivant sont libellées différemment selon qu'il s'agit du clavier de la France, du clavier du Canada ou du clavier des États-Unis. Reportez-vous à ce tableau pour faire correspondre les touches françaises figurant dans le présent document aux touches de votre clavier.

France	Canada	Etats-Unis
↖ (Pos1)	↖	Home
Fin	Fin	End
⇞ (PgAr)	⇞	PgUp
⇟ (PgAv)	⇟	PgDn
Inser	Inser	Ins
Suppr	Suppr	Del
Echap	Echap	Esc
Attn	Intrp	Break
Impr écran	ImpEc	PrtSc
Verr num	Num	Num Lock
Arrêt défil	Défil	Scroll Lock
(Verr maj)	FixMaj	Caps Lock
AltGr	AltCar	Alt (à droite)

Brevets

Il est possible qu'IBM détienne des brevets ou qu'elle ait déposé des demandes de brevets portant sur certains sujets abordés dans ce document. Le fait qu'IBM vous fournisse le présent document ne signifie pas qu'elle vous accorde un permis d'utilisation de ces brevets. Vous pouvez envoyer, par écrit, vos demandes de renseignements relatives aux permis d'utilisation au directeur général des relations commerciales d'IBM, 3600 Steeles Avenue East, Markham, Ontario, L3R 9Z7.

Assistance téléphonique

Si vous avez besoin d'assistance ou si vous voulez commander du matériel, des logiciels et des publications IBM, contactez IBM direct au 1 800 465-1234.

Chapitre 1. Présentation de Prerequisite Scanner

IBM® Prerequisite Scanner est un outil d'analyse qui assure l'identification, le contrôle et la vérification des prérequis pour le logiciel spécifié avant le déploiement réel. Il analyse les prérequis matériels et logiciels en fonction des valeurs définies pour les propriétés de prérequis. Le scanner affiche les résultats de l'analyse dans l'interface de ligne de commande et enregistre également les résultats sous les fichiers texte et éventuellement les fichiers XML. Il écrit également des messages d'information, de trace et de débogage dans les fichiers journaux.

Prerequisite Scanner vérifie le système d'exploitation de la machine et s'il s'agit de la version correcte pour le logiciel spécifié. Si l'un des contrôles individuels des prérequis échoue, l'analyse globale échoue.

Vous pouvez exécuter Prerequisite Scanner après une installation ou à tout moment afin de confirmer votre environnement en cours. Prerequisite Scanner ne nécessite pas l'exécution du programme d'installation du logiciel dont vous souhaitez contrôler les prérequis.

Vous pouvez étendre Prerequisite Scanner pour rechercher des prérequis ne faisant pas partie de l'ensemble principal des contrôles de prérequis fournis avec le scanner.

Prerequisite Scanner appelle les types de scripts suivants en fonction de votre plateforme :

- Windows : VBScript et par lots
- UNIX : interpréteur de commandes

Remarque : Vous ne pouvez pas exécuter les scripts UNIX sous les systèmes Windows même si vous avez installé un environnement de type UNIX sur les machines Windows, par exemple Cygwin.

Architecture Prerequisite Scanner

IBM Prerequisite Scanner comprend les principaux éléments suivants : un script à exécuter dans une interface de ligne de commande, un ensemble de propriétés pour les contrôle prérequis, des fichiers de configuration de propriété de prérequis, de collecteurs prérequis et des évaluateurs prérequis. Les résultats de l'exécution de Prerequisite Scanner sont disponibles dans différents formats de sortie.

Propriétés de prérequis

Les propriétés de prérequis correspondent aux valeurs attendues pour différents prérequis logiciels et matériels exigés par les produits ou solutions à installer. Les exemples de propriétés de prérequis comprennent l'espace disque total disponible sur la machine, l'ensemble de ports non utilisés sur une machine et l'ensemble actuel d'applications installées.

Les valeurs de ces propriétés de prérequis pouvant changer avec différents produits, les propriétés et leurs valeurs sont représentées sous forme de paires de valeurs de nom avec des qualificatifs facultatifs. Elles sont contenues dans les fichiers de configuration des propriétés de prérequis. Il y a une seule propriété de prérequis sur chaque ligne.

Les propriétés des prérequis sont conformes au format suivant :

```
[prefix_identifier.]property_name[.suffix_identifier]=
[[qualifier_name :qualifier_value]]property_value
```

où :

- *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis, comme indiqué dans tableau 3, à la page 4. Cet identificateur à préfixe est obligatoire pour certaines des catégories prédéfinies.
- *property_name* est le nom de la propriété de prérequis.
- *suffix_identifier* est un identificateur facultatif pour un sous-type de propriétés de prérequis, comme indiqué dans tableau 4, à la page 7.
- *qualifier_name* est un attribut facultatif pour la propriété de prérequis. IBM Prerequisite Scanner l'utilise pour définir la propriété de prérequis ou le type de contrôle à effectuer sur la propriété de prérequis.

  Remarque : Vous pouvez avoir plusieurs qualificatifs, chacun séparé par une virgule. L'ensemble de qualificatifs doit être placé entre crochets `[]`.
- *qualifier_value* est la valeur de l'attribut facultatif. Chaque qualificatif et sa valeur doivent être délimités par deux points `:`.
- *property_value* est la valeur de la propriété de prérequis et elle peut correspondre à une chaîne ou un entier.

Une propriété de prérequis peut contenir une ou plusieurs valeurs suivant le type de données et le qualificatif, comme indiqué ci-dessous :

- Un entier unique, par exemple `8080` permettant de représenter la valeur d'un numéro de port.
- Une gamme ou groupe d'entiers représentés par des caractères spéciaux, comme indiqué dans tableau 1.

Tableau 1. Caractères spéciaux permettant de représenter les types de gammes

Caractères spéciaux	Description
`*`	Identifie une marque de réservation pour plusieurs valeurs. Par exemple, `ports.*` peut représenter un sur-ensemble de ports à la fois pour un produit de base de données `ports.DB` et IBM WebSphere Application Server, `ports.WAS`.
`+`	Identifies that the actual version must at least match the value for expected version. For example, `os.versionNumber=5.0+`, means that the version must be 5.0 or later.
`-`	Indique que la version réelle doit au mieux correspondre à la valeur de la version attendue. Par exemple, `os.versionNumber=5.0-` signifie que la version doit être 5.0 ou antérieure.
`.*`	Indique que la version réelle doit correspondre à une valeur de caractère générique pour la version attendue. Exemple : `os.versionNumber=5.*, means that the version can be 5.0, 5.0.1 or 5.5.`

Restriction : Sous les systèmes Windows, le caractère générique `*` est uniquement pris en charge s'il est utilisé dans une expression régulière de la propriété de prérequis `OS Version`.

- Chaîne pouvant représenter l'une des valeurs suivantes pour les types de prérequis :
  - Valeur numérique avec une unité, par exemple `8GB` ou `10MB`
  - Application, système d'exploitation, architecture ou module, par exemple `IBM Lotus Symphony, RedHat Enterprise Linux 5.4, 32-bit` ou `ftp`

    Remarque : Une chaîne peut également comprendre plusieurs valeurs séparées par une virgule, par exemple une liste d'applications.
  - L'une des valeurs représentées par l'une des combinaisons suivantes telles que `True|False, Available|Unavailable` ou `Enabled|Disabled`

tableau 2 présente des exemples de propriétés de prérequis.

Tableau 2. Exemples de propriétés de prérequis

Propriété de prérequis	Explication
`Disk=1GB`	Volume d'espace disque libre, où : • *property_name* est `Disk` • *property_value* est `1GB`
`user.isAdmin=True`	Indique si l'utilisateur connecté appartient à un groupe administrateur, où : • *prefix_identifier* est `user` pour les propriétés de prérequis de l'utilisateur • *property_name* est `isAdmin` • *property_value* est `True`

`network.availablePorts.DB=60000-60005`
`network.availablePorts.WAS=8080`
`network.availablePorts.FTP=21`
	Vérifie que les ports 60000-60005 sont disponibles pour le serveur de base de données, le port 8080 est disponible pour WebSphere Application Server et le port 21 pour FTP, où :
	• *prefix_identifier* est `network` pour les propriétés de prérequis générales
	• *property_name* est `availablePorts`
	• *suffix_identifier* sont `DB` pour les ports de base de données disponibles, `WAS` pour le port WebSphere Application Server disponible et FTP pour le port `FTP` disponible
	• *property_value* est `60000-60005`, `8080` ou `21`

`os.dir.home=[dir:/home,type:permission]755+`	Vérifie que le répertoire initial dispose des autorisations `drwxr-xr-x`, où :
	• *prefix_identifier* est `os` pour les propriétés de prérequis du système d'exploitation
	• *property_name* est `dir`
	• *suffix_identifier* est `home` pour le répertoire à contrôler
	• *qualifier_name* sont `dir` et
	`type that qualify the prerequisite property and type of check`
	• *qualifier_value* sont `home` et
	`permission, the values for the qualifiers`
	• *property_value* est `755+`, c'est-à-dire la représentation de chiffre octal des autorisations d'accès pour le répertoire initial

Vous pouvez ajouter ou modifier les propriétés de prérequis prédéfinies pour chaque produit pour lequel vous souhaitez exécuter Prerequisite Scanner. Vous pouvez également créer des propriétés de prérequis personnalisées et utiliser des collecteurs et des évaluateurs Prerequisite Scanner comme exigé afin d'analyser et de comparer les propriétés de prérequis.

Concepts associés :

«Catégories prédéfinies des propriétés de prérequis», à la page 9
IBM Prerequisite Scanner fournit un ensemble de qualifiants de base pour certaines propriétés de prérequis d'une catégorie prédéfinie. Les qualifiants représentent les attributs de la propriété de prérequis que Prerequisite Scanner utilise pour définir la propriété de prérequis ou le type de contrôle à effectuer sur la propriété de prérequis.

Catégories prédéfinies des propriétés de prérequis

IBM Prerequisite Scanner fournit un ensemble de propriétés de prérequis de base pour différentes catégories de données : commune, logiciel installé, système d'exploitation, utilisateur, connectivité, Internet Explorer, serveur de base de données, variables d'environnement et réseau comprenant des propriétés spécifiques à la plateforme pour Windows et UNIX.

<prefix_identifier> est un identificateur pour une catégorie prédéfinie de propriétés de prérequis.

tableau 3 présente les catégories prédéfinies des prérequis matériels et logiciels.

Tableau 3. Catégories de propriétés de prérequis de base

Catégorie de données	Description	Identificateur à préfixe obligatoire
Commune	Cette catégorie vérifie les prérequis communs tels que la vitesse du processeur, la mémoire vive, le disque et l'espace temporaire. Cet exemple présente la propriété de prérequis permettant de vérifier le système d'exploitation : `OS Version=RedHat Enterprise Linux 5.4`	Aucune
Logiciel installé	Cette catégorie vérifie les prérequis du logiciel installé tels que les programmes enregistrés dans le registre Windows et que cygwin et gskit sont installés. Cet exemple présente la propriété de prérequis permettant d'analyser le registre de système d'exploitation des programmes installés avec les emplacements : `installedSoftware=`*`list_of_installed_programs`	Aucune
Utilisateur	Cette catégorie vérifie les prérequis de l'utilisateur, par exemple que l'utilisateur connecté dispose des droits d'administration ou est le superutilisateur. Cet exemple présente la propriété de prérequis permettant de vérifier que l'utilisateur connecté est membre du groupe administrateur : `user.isAdmin=True`	`user`
Système d'exploitation	Cette catégorie vérifie les prérequis du système d'exploitation tels que la version, l'architecture, la mémoire totale, la mémoire disponible et la mémoire physique totale. Cet exemple présente la propriété de prérequis permettant de vérifier que le service du registre distant fonctionne : `os.isServiceRunning.remoteRegistry=True`	`os`
Connectivité	Cette catégorie vérifie les prérequis de connectivité, par exemple que Telnet est en cours d'exécution et à quelles adresses IP et quels ports le scanner peut être connecté.	Aucune

Tableau 3. Catégories de propriétés de prérequis de base (suite)

Catégorie de données	Description	Identificateur à préfixe obligatoire
Réseau	Cette catégorie vérifie les prérequis de réseau qui peuvent être communs à toutes les plateformes, par exemple s'il y a des ports disponibles. Cet exemple présente la propriété de prérequis permettant de vérifier que le port `8080` est disponible pour IBM WebSphere Application Server : `network.availablePorts.was=8080`	`network`
Réseau Windows	Cette catégorie vérifie les prérequis du réseau Windows, par exemple que NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING. Cet exemple présente la propriété de prérequis permettant de vérifie qu'au moins un adaptateur d'une adresse IP valide dispose de NetBIOS activé comme protocole : `network.netBIOSEnabled=True`	`network`
Réseau UNIX	Cette catégorie vérifie les prérequis du réseau UNIX, par exemple que NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING. Cet exemple présente la propriété de prérequis permettant de vérifier que le système hôte local répond au protocole de la commande PING : `network.pingLocalhost=True`	`network`
Internet Explorer	Cette catégorie vérifie les prérequis de Microsoft Internet Explorer telles que la version. Cet exemple présente la propriété de prérequis permettant de vérifier que la version d'Internet Explorer est `7.0` : `internetExplorer.version=7.0`	`internetExplorer`
Serveur de base de données DB2	Cette catégorie vérifie les prérequis de DB2 tels que la version. Cet exemple présente la propriété de prérequis permettant de vérifier que la version de DB2 est au moins `9.5` : `DB2 Version=9.5.*`	`DB2`
Serveur de base de données Oracle	Cette catégorie vérifie les prérequis d'Oracle tels que la version. Cet exemple présente la propriété de prérequis permettant de vérifier que la version de client Oracle est au moins `9.2.0.8` : `oracle.Client=9.2.0.8+`	`Oracle`
Variables d'environnement	Cette catégorie vérifie les prérequis des variables d'environnement, par exemple que la variable d'environnement a été définie. Cet exemple présente la propriété de prérequis permettant de vérifier que le chemin d'accès aux classes contient le fichier JAR Derby : `env.classpath.derbyJAR=False`	`env`
Autonomic Deployment Engine	Cette catégorie vérifie les prérequis de Autonomic Deployment Engine, par exemple que Autonomic Deployment Engine est installé ou qu'il s'agit de l'unité d'installation de Tivoli Integrated Portal. Cet exemple présente la propriété de prérequis permettant de vérifier que l'unité d'installation de Tivoli Integrated Portal version 2.1.1.0 ou 2.1.1.1 est installée sur un système Windows : `de.installationUnit=regex{.*C37109911C8A11D98E1700061BDE7AEA.*|.*TIP 2.1.1.0.*|.*TIP 2.1.1.1.*}`	`de`
Serveur de base de données MS SQL	Cette catégorie vérifie les prérequis de SQL MS tels que la version. Cet exemple présente la propriété de prérequis permettant de vérifier que la version de serveur SQL MS est SQL Server 2008 R2 Developer Edition : `mssql.Server=10.50.1600.1`	`mssql`

Sous-types prédéfinis des propriétés de conditions prérequises

IBM Prerequisite Scanner fournit un ensemble de sous-types de base pour certaines propriétés de conditions prérequises d'une catégorie prédéfinie. Les sous-types continuent à classer par catégories un propriété de conditions prérequises telle qu'une catégorisation par application, par utilitaire ou par sous-type de service.

Par exemple, vous pouvez avoir une propriété de conditions prérequises pour les ports réseau disponibles. Vous pouvez encore classer par catégories cette propriété de conditions prérequises pour contrôler les ports disponibles d'un serveur de base de données, d'un serveur d'application ou d'un protocole.

<suffix_identifier> est un identificateur facultatif pour un sous-type de nom de propriété de conditions prérequises.

Le tableau 4 présente les sous-types prédéfinis pour différentes catégories de propriétés de conditions prérequises comprenant <*suffix_identifier*>.

Tableau 4. Sous-types prédéfinis

Sous-type de propriété de conditions prérequises	Identificateur à suffixe	Plateforme	Description	Valeurs valides pour le sous-type
Catégorie de réseau indépendante de la plateforme				
`network.availablePorts.`*`app_type`*	*app_type*	Tous	Utilisez cette convention d'attribution de nom pour vérifier que le port ou la gamme de ports n'est pas en mode écoute ou est disponible pour le type d'application *app_type*.	Chaîne représentant *app_type*, par exemple : • `DB2` contrôle les ports du serveur de base de données DB2 • `WAS` contrôle les ports de WebSphere Application Server • `ftp` contrôle le port FTP
`network.portsInUse.`*`app_type`*	*app_type*	Tous	Utilisez cette convention d'attribution de nom pour vérifier que le port ou la gamme de ports est en mode écoute ou utilisée pour le type d'application *app_type*.	Chaîne représentant *app_type*, par exemple : • `DB2` contrôle les ports du serveur de base de données DB2 • `WAS` contrôle les ports de WebSphere Application Server • `ftp` contrôle le port FTP
Catégorie du système d'exploitation				
`os.dir.`*`dir_name`*	*dir_name*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier le système de fichiers *dir_name*. La valeur de la propriété de prérequis utilise les qualificatifs prédéfinis.	Chaîne représentant *dir_name*, par exemple : • `tmp` • `home`
`os.file.`*`script_name`*	*script_name*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier que le script *script_name* est disponible sur la machine.	Chaîne représentant *script_name*, par exemple : • `bash` • `expect` • `gzip` • `tar`

`os.`
`isService`
`Exécution en cours.`
`service_name`
	service_name	Windows	Utilisez cette convention d'attribution de nom pour vérifier que le service *service_name* est en cours d'exécution sur la machine.	Chaîne représentant *service_name*, par exemple :
				• `remoteRegistry`
				• `DNSClient`
				• `terminalServices`

Tableau 4. Sous-types prédéfinis (suite)

Sous-type de propriété de conditions prérequises	Identificateur à suffixe	Plateforme	Description	Valeurs valides pour le sous-type
`os.lib.` *`lib_name_version`*	*lib_name _version*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier que la version prise en charge de la bibliothèque *lib_name_version* est installée sur la machine.	Chaîne permettant de représenter *lib_name_version*, par exemple, en gras : • bibliothèque **`libstdc++.so.#`** 32 bits • bibliothèque **`libstdc++.so.#`** 64 bits • bibliothèque **`libXft.so.#`** 32 bits • bibliothèque **`libXtst.so.#`** 32 bits • bibliothèque **`libaio.so.#`** 64 bits • niveau d'exécution XLC **`xlC.rte`** 32 bits • exécution XLC **`xlC.aix50.rte`** 32 bits pour AIX version 5.3 • exécution XLC **`xlC.aix61.rte`** 32 bits pour AIX version 6.1 • bibliothèque AIX IOCP **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, ensemble de fichiers de code ISO pour le système d'exploitation de base AIX `regex {`*`str`*`}`, expression régulière avec le paramètre d'entrée *str* représentant le motif recherché pour le nom de la bibliothèque, par exemple : `regex {.*libgcc.*}` Vérifie si une version de la bibliothèque d'exécution de bas niveau GCC `libgcc` pour ce système d'exploitation existe.

Tableau 4. Sous-types prédéfinis (suite)

Sous-type de propriété de conditions prérequises	Identificateur à suffixe	Plateforme	Description	Valeurs valides pour le sous-type
`os.package.` *`package_name`*	*package_name*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier que la version prise en charge du module *package_name* est installée sur la machine.	Chaîne permettant de représenter *package_name*, par exemple, en gras : • interpréteur de commandes **bash** • **expect** pour le module d'extension TCL • **libgcc** pour le module d'exécution de bas niveau GCC • **openssh** pour interpréteur de commandes sécurisé Open Source • **openssl** pour le kit d'outils Open Source de SSL/TLS • **perl** pour le module de script Perl • **rpm** pour le RPM ou les modules de génération RPM • **telnet** pour le module Telnet • **wget** pour le module de récupération de fichiers GNU
`os.space.` *`dir_name`*	*dir_name*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier l'espace disque disponible pour le système de fichiers *dir_name*. La valeur de la propriété de prérequis utilise les qualificatifs prédéfinis.	Chaîne représentant *dir_name*, par exemple : • `usr` • `home` • `tmp` • `var`

Catégories prédéfinies des propriétés de prérequis

IBM Prerequisite Scanner fournit un ensemble de qualifiants de base pour certaines propriétés de prérequis d'une catégorie prédéfinie. Les qualifiants représentent les attributs de la propriété de prérequis que Prerequisite Scanner utilise pour définir la propriété de prérequis ou le type de contrôle à effectuer sur la propriété de prérequis.

Par exemple, vous pouvez avoir une propriété de prérequis pour un système de fichiers. Vous pouvez indiquer quel contrôle effectuer pour cette propriété de prérequis en vous basant sur le nom du système de fichiers et les attributs d'autorisations d'accès. Vous pouvez également indiquer quel type d'unités utiliser lors du contrôle de l'espace disque disponible en vous basant sur le chemin d'accès au système de fichiers et les attributs d'unité.

Les qualifiants prennent en charge la personnalisation pour répondre aux besoins de votre environnement et empêcher le scanner d'avoir à émettre des hypothèses

implicites concernant les attributs des prérequis multidimensionnels comme le chemin d'accès par défaut et les autorisations d'accès. Vous pouvez changer les valeurs des qualifiants prédéfinis, mais vous ne pouvez pas ajouter de nouveaux qualificatifs à l'ensemble existant de qualificatifs prédéfinis pour une propriété de prérequis prédéfinie.

Les qualificatifs doivent être conformes au format suivant :

```
[qualifier_name :qualifier_value, qualifier_name :qualifier_value]
property_value
```

où :

- *qualifier_name* est un attribut facultatif pour la propriété de prérequis que IBM Prerequisite Scanner utilise pour indiquer la propriété de prérequis ou le type de contrôle à effectuer sur la propriété de prérequis.
- *qualifier_value* est la valeur de l'attribut facultatif.

  La valeur du qualifiant peut également être une paire de valeurs de nom permettant de prendre en charge plusieurs valeurs valides suivante le type d'utilisateur. Par exemple, différents chemins pour le répertoire initial suivant que l'utilisateur soit un superutilisateur ou non.
- *property_value* est la valeur de la propriété de prérequis et elle peut correspondre à une chaîne ou un entier.

Chaque qualificatif et sa valeur doivent être délimités par deux points `:`. Vous pouvez avoir plusieurs qualificatifs, chacun séparé par une virgule. L'ensemble de qualificatifs doit être placé entre crochets `[]`.

tableau 5 présente les qualificatifs prédéfinis pour différentes catégories de propriétés de prérequis. Certaines propriétés de prérequis utilisent également des sous-types prédéfinis pour continuer à classer par catégories une propriété de prérequis.

Important : Vous ne pouvez pas utiliser les qualificatifs prédéfinis avec d'autres propriétés de prérequis prédéfinies.

Tableau 5. Qualificatifs prédéfinis

Propriété de prérequis	Plateforme	Description	Qualificatifs et valeurs valides
Catégorie de système d'exploitation avec un sous-type prédéfini			

`os.dir.`*`dir_name`*	UNIX	Vérifie le système de fichiers *dir_name* en fonction des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le système de fichiers à contrôler
		• attribut `type` permettant de déterminer l'attribut de système de fichiers à contrôler, par exemple la représentation numérique octale `<octal_digits>` pour les autorisations d'accès à ce système de fichiers
		
		<dir_name> peut représenter par exemple :
		• `tmp`
		• `home`
			Chaîne présentant le format de qualifiant suivant :
			`[dir:`*`dir_name`*`, type:permission]`
			`octal_digits``+`
			
			Par exemple, pour vérifier que le répertoire initial dispose des autorisations `drwxr-xr-x` :
			`os.dir.home=[dir:/home, type:permission]755+`

Tableau 5. Qualificatifs prédéfinis (suite)

Propriété de prérequis	Plateforme	Description	Qualificatifs et valeurs valides

`os.space.`
`dir_name`
	UNIX	Vérifie l'espace disque disponible pour le système de fichiers *dir_name* spécifié en se basant sur un ou plusieurs des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au système de fichiers à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
		
		La valeur de l'attribut `dir` dépend de l'utilisateur connecté ; ainsi, la valeur est une paire de valeurs de nom permettant de représenter le type d'utilisateur, à savoir superutilisateur ou non, et le chemin d'accès associé.
		
		dir_name peut représenter par exemple :
		• `usr`
		• `home`
		• `tmp`
		• `var`
			Chaîne présentant le format de qualificatif suivant pour le système de fichiers d'un superutilisateur :
			`[dir:`**`root`**`=`*`dir_path`*`,`
			`unit:`*`unit_name`*`]`
			`disk_space`
			
			Par exemple :
			`os.space.usr=`
			`[dir:root=/usr/ibm/common/acsi,`
			`unit:GB]200`
			
			Chaîne présentant le format de qualificatif suivant pour le système de fichiers d'un utilisateur non superutilisateur :
			`[dir:`**`non_root`**`=`*`dir_path`*`,`
			`unité :`*`unit_name`*`]`
			`disk_space`
			
			Par exemple :
			`os.space.home=`
			`[dir:non_root=USERHOME/`
			`.acsi_HOST,`
			`unit:MB]200`
			
			Chaîne présentant le format de qualificatif suivant utilisant un seul qualificatif :
			`[dir:`*`dir_path`*`]`
			`disk_space` `MB`
			
			Par exemple :
			`os.space.home=[dir:/home/sat]250MB`

Catégorie de système d'exploitation sans sous-type prédéfini			

`os.mountcheck`	UNIX	Vérifie que le système de fichiers est installé en fonction des attributs de qualification suivants :
		• attribut `drive` permettant de déterminer quel répertoire correspond au système de fichiers installé
		• attribut `nosuid` permettant de déterminer si l'option d'installation est paramétrée si le système de fichiers est installé
			Chaîne présentant le format de qualificatif suivant :
			`[drive:`*`dir_name`*`,`
			`mount_option` `:`
			`false|true]`
			`True|False`
			
			Par exemple, pour vérifier que le répertoire `/home` est installé et que l'option `nosuid` n'est pas paramétrée :
			`os.mountcheck=[drive:/home,`
			`nosuid:false]True`

Tableau 5. Qualificatifs prédéfinis (suite)

Propriété de prérequis	Plateforme	Description	Qualificatifs et valeurs valides

`os.SELinux`	Linux	Vérifie le statut d'application de la fonction Linux d'application de sécurité en se basant sur les attributs de qualification suivants :
		• attribut `source` permettant de déterminer la commande à utiliser pour le système d'exploitation associé
			• Chaîne présentant le format de qualificatif suivant :
			`[source:Command] Disabled|Enabled`
			Par exemple, pour vérifier que la fonction est désactivée ou présente un statut autorisé sur le système d'exploitation Red Hat ou SUSE :
			`os.SELinux=[source: Command]Disabled`
			• Chaîne sans qualificatif dans laquelle le système d'exploitation est une variante générique Linux :
			`os.SELinux=Disabled`

`os.ulimit`	UNIX	Utilisez cette convention de dénomination pour vérifier qu'un nombre illimité de processus peut être exécuté, en se basant sur les attributs de qualification suivants :
		• attribut `type` permettant de déterminer quel attribut supplémentaire contrôler, par exemple `filedescriptorlimit` vérifie la limite du nombre de descripteurs de fichiers que les processus peuvent ouvrir
			Chaîne présentant le format de qualificatif suivant :
			`[type:`*`limit_name`*`]`*`limit_value`*`, limited|unlimited`
			Par exemple, pour vérifier que le descripteur de fichier est supérieur à 8192, avec un nombre illimité de processus :
			`os.ulimit= [type:filedescriptorlimit] 8192+,unlimited`
			Voici les types valides de limites à vérifier, *limit_name* représentant le type de limite :
			• `ALL` vérifie toutes les limites
			• `corefilesizelimit`
			• `datasegmentlimit`
			• `filedescriptorlimit`
			• `filesizelimit`
			• `hardlimit`
			• `processlimit`
			• `maxmemorysizelimit`
			• `maxprocesseslimit`
			• `stacksizelimit`
			• `threadlimit`

Tableau 5. Qualificatifs prédéfinis (suite)

Propriété de prérequis	Plateforme	Description	Qualificatifs et valeurs valides
Catégorie commune sans sous-type prédéfini			

`Disk`	Windows	Volume d'espace disque libre avec les attributs de qualification facultatifs suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au répertoire à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
			Chaîne présentant le format de qualificatif suivant :
			`[dir:`*`dir_path`*`, unité :`*`unit_name`*`]`
			`disk_space`
			Par exemple :
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			Format numérique en Mo ou Go :
			`<disk_space>MB|GB`
			Par exemple :
			`Disk=250MB`

Codes produit

IBM Prerequisite Scanner utilise des codes multicaractères dans les noms de fichier et les noms de paramètre afin d'identifier des produits et des composants et détermine quel type de fichier de configuration il faut utiliser.

product_code

Il s'agit de la variable représentant un code produit sur les systèmes Windows ou UNIX. Les codes produit identifient le produit, une plateforme individuelle, telle que Windows, AIX, HP-UX, Linux et Solaris, et éventuellement la version du système d'exploitation prise en charge par ce produit. Ils sont stockés dans le fichier `codename.cfg`. Tout produit prenant en charge plusieurs plateformes comprend plusieurs codes produit, chacun identifiant un produit, une plateforme et la version du système d'exploitation, selon les besoins.

Par exemple, les codes produit`COD`, et `COK` et `COX` identifient certains des systèmes d'exploitation pris en charge et les versions pour IBM Tivoli Provisioning Manager :

```
COD=Tivoli Provisioning Manager for
AIX 6.1
COK=Tivoli Provisioning Manager for HP-UX
COX=Tivoli Provisioning Manager for Windows 2008
```

Lorsque vous exécutez Prerequisite Scanner, vous entrez le code du produit et éventuellement la version du produit comme paramètres d'entrée. Le scanner vérifie si le code produit existe dans le fichier `codename.cfg`. Sur les systèmes UNIX, le scanner se ferme s'il ne trouve pas le code. Sur les systèmes Windows, ce n'est pas le cas :

s'il ne trouve pas le code, le scanner utilise ensuite les paramètres d'entrée pour trouver le fichier de configuration dans le répertoire *ips_root*`/Windows|UNIX_Linux`. Le nom du fichier contient le même code produit et la même version du produit que les paramètres d'entrée. Si vous n'entrez pas le paramètre facultatif de la version du produit, le scanner utilise alors la version la plus récente du fichier de configuration qu'il trouvera dans le répertoire. Prerequisite Scanner commence alors le scan.

Remarque : Sur les systèmes Windows uniquement : si le code produit n'existe pas dans le fichier `codename.cfg` mais qu'un fichier de configuration dont le nom contient le code produit existe, Prerequisite Scanner affiche le code produit et le numéro de version dans la sortie, sans définir le nom du produit.

Prerequisite Scanner fichiers de configuration

Les IBM Prerequisite Scanner fichiers de configuration pour les plates-formes individuelles contiennent les propriétés pré-requises et les valeurs attendues pour chaque plate-forme prise en charge par le produit. Prerequisite Scanner fournit une gamme prédéfinie de fichiers de configuration que vous pouvez modifier. Pour prendre en charge de nouveaux produits et de nouvelles plates-formes vous devez créer des fichiers de configuration.

Les fichiers de configuration ont une extension de fichiers `.cfg`. Vous les stockez dans le répertoire*ips_root*/*<OS>*, où *<OS>* est le nom du type de système d'exploitation, par exemple, `Windows` ou `UNIX_Linux`.

Les fichiers de configuration doivent s'inscrire sur le registre des règles suivantes :

- l'extension du fichier doit être `.cfg`
- Convention d'attribution de noms pour le nom du fichier :

  product_code[_*<version>*].cfg

  où :
  - *product_code*

    Il s'agit de la variable représentant un code produit sur les systèmes Windows ou UNIX. Les codes produit identifient le produit, une plateforme individuelle, telle que Windows, AIX, HP-UX, Linux et Solaris, et éventuellement la version du système d'exploitation prise en charge par ce produit. Ils sont stockés dans le fichier `codename.cfg`. Tout produit prenant en charge plusieurs plateformes comprend plusieurs codes produit, chacun identifiant un produit, une plateforme et la version du système d'exploitation, selon les besoins.
  - *<version>* est le code à 8 chiffres représentant la version, l'édition, la modification et le niveau avec deux chiffres pour chaque partie du code. Par exemple : 7.3.21 est 07032100.
- Les groupes de propriétés de prérequis sous des sections qui doivent suivre une convention d'attribution de noms pour les titres de section.
- Un format standard pour chaque propriété de prérequis est une paire de valeurs nom avec des qualificateurs optionnels et une seule propriété pour chaque ligne :

  [*<prefix_identifier>*.]*<property_name>*[.*<suffix_identifier>*]=
  [[*<qualifier_name>* :*<qualifier_value>*]]*<property_value>*

Exemple de fichier de configuration sans sections

Cette exemple vérifie les propriétés de prérequis mais ne se différencie pas des propriétés de prérequis pour les versions de système d'exploitation requises.

```
os.space.var=[dir:root=/var/ibm/common/acsi,unit:MB]1.0
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:MB]200
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
os.space.tmp=30MB
env.classpath.derbyJAR=False
network.pingSelf=True
```

```
network.pingLocalhost=True
network.availablePorts.Derby=4130
OS Version=RedHat Enterprise Linux 4.*,RedHat Enterprise Linux 5.*
os.package.compat-libstdc++-33=compat_libstdc++_33
os.package.libgcc=libgcc-3.4.3-9
```

Concepts associés :

«Sections des fichiers de configuration»
Les propriétés de prérequis peuvent être regroupées dans un ensemble de sections dans les fichiers de configuration, chaque section représentant une catégorie de type de données. Les sections sont facultatives dans les fichiers de configuration.

Sections des fichiers de configuration

Les propriétés de prérequis peuvent être regroupées dans un ensemble de sections dans les fichiers de configuration, chaque section représentant une catégorie de type de données. Les sections sont facultatives dans les fichiers de configuration.

La convention de dénomination du titre de sections est la suivante :

[*category_name* :*category_value*]

où :

- *category_name* est le code à caractères multiples qui représente la catégorie de type de données
- *category_value* est le code à caractères multiples qui représente une valeur autorisée pour la catégorie

  Remarque : Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

Chaque nom de catégorie et sa valeur doivent être délimités par

`: colon and enclosed by [] square brackets.`

Vous pouvez avoir plusieurs catégories de types de données en combinant les titres de section et en limitant ainsi les propriétés de prérequis uniquement à cet ensemble de catégories spécifié.

[*category_name* :*category_value*][*category_name* :*category_value*]

Par exemple, pour spécifier les propriétés de prérequis s'appliquant à une machine exécutant un système d'exploitation 32 bits SUSE Linux Enterprise Server version 11, Itanium :

`[OSType:SUSELinuxEnterpriseServer11][OSArch:64-bit][CPU:Itanium]`

Pour toutes les plateformes, vous pouvez utiliser le symbole | logique OR pour utiliser l'une des catégories de type de données. Par exemple, pour avoir l'une des variables d'environnement paramétrée sur `True`, la combinaison de titres de sections est la suivante :

- Systèmes **UNIX**

  `[@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True|@TPAE_J2EE_FEATURE:True]`
- Systèmes **Windows**

  `[@TPAE_DB_FEATURE:True]|[@TPAE_DIR_FEATURE:True]|[@TPAE_J2EE_FEATURE:True]`

Important : La position du symbole | logique OR est différente entre les systèmes Windows et UNIX. Pour les systèmes UNIX, l'ensemble des titres de section est contenu dans un ensemble de crochets `[]` uniquement avec chaque titre de section séparé par le symbole. Pour les systèmes Windows, le symbole délimite chaque titre de section complet avec les crochets `[]` associés.

Pour les systèmes Windows uniquement, vous pouvez utiliser le symbole `!` symbole logique NOT pour exclure une catégorie de type de données. Par exemple, pour exclure une variante Windows Server 2003 R2, la combinaison de titres de section est la suivante : `[OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]`

tableau 6, à la page 17 présente les catégories de type de données prises en charge et les valeurs autorisées associées.

Tableau 6. Catégories de type de données et valeurs prises en charge

Catégorie de type de données	Description	Valeurs autorisées

`OSType`	Type de système d'exploitation	• UNIX
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les plateformes UNIX, y compris AIX, HP-UX, Linux et Solaris, par exemple :
		`[OSType:UNIX]`
		• AIX
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation AIX, par exemple :
		`[OSType:AIX]`
		• HP-UX
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation HP-UX, par exemple :
		`[OSType:HP-UX]`
		• LINUX
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Linux, par exemple :
		`[OSType:LINUX]`
		• RedHat
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation RedHat Linux, par exemple :
		`[OSType:RedHat]`
		• RedHatEnterpriseLinuxServer
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation du serveur RedHat Enterprise Linux, par exemple :
		`[OSType:RedHatEnterpriseLinuxServer]`
		• SUSE
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation SUSE Linux, par exemple :
		`[OSType:SUSE]`
		• SUSELinuxEnterpriseServer
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation SUSE Linux Enterprise Server, par exemple :
		`[OSType:SUSELinuxEnterpriseServer]`
		• Solaris
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Solaris, par exemple :
		`[OSType:Solaris]`

Tableau 6. Catégories de type de données et valeurs prises en charge (suite)

Catégorie de type de données	Description	Valeurs autorisées

		• Windows
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes des systèmes d'exploitation Windows, par exemple :
		`[OSType:Windows]`
		• Windows 2000 Workstation (version 5.0.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows 2000, par exemple :
		`[OSType:Windows 2000]`
		• Windows XP Workstation (version 5.1.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows XP Professional 32 bits, par exemple :
		`[OSType:Windows XP]`
		• Windows XP Workstation (version 5.2.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows XP Professional 64 bits, par exemple :
		`[OSType:Windows XP]`
		• Windows Vista Workstation (version 6.0.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows Vista, par exemple :
		`[OSType:Windows Vista]`
		• Windows 7 Workstation (version 6.1.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows 7, par exemple :
		`[OSType:Windows 7]`
		• Windows 2000 Server (version 5.0.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows 2000 Server, par exemple :
		`[OSType:Windows 2000]`
		• Windows Server 2003 (version 5.2.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows Server 2003, par exemple :
		`[OSType:Windows Server 2003]`
		• Windows Server 2003 R2 (version 5.2.* et autre type de description de système d'exploitation R2)
		Indique que toutes les propriétés de cette catégorie sont communes uniquement à la variante du système d'exploitation Windows Server 2003 R2, par exemple :
		`[OSType:Windows Server 2003 R2]`

Tableau 6. Catégories de type de données et valeurs prises en charge (suite)

Catégorie de type de données	Description	Valeurs autorisées

		• Windows Server 2008 (version 6.0.*)
		Indique que toutes les propriétés de cette catégorie sont communes à toutes les variantes du système d'exploitation Windows Server 2008, par exemple :
		`[OSType:Windows Server 2008]`
		• Windows Server 2008 R2 (version 6.1.*)
		Indique que toutes les propriétés de cette catégorie sont communes uniquement à la variante du système d'exploitation Windows Server 2008 R2, par exemple :
		`[OSType:Windows Server 2008 R2]`
		• *<OS_Name_Version>*
		Indique que toutes les propriétés de cette catégorie sont communes à cette version du système d'exploitation, par exemple :
		`[OSType:RedHatEnterpriseLinuxServer4.2]`
		
		Remarque : Le caractère générique spécial `*`, est autorisé à définir plusieurs versions.

`OSArch`	Architecture du système d'exploitation	• 32 bits, par exemple :
		`[OSArch:32-bit]`
		• 64 bits, par exemple :
		`[OSArch:64-bit]`

`Unité centrale`	Nom de la famille de processeurs génériques	Itanium, par exemple : `[CPU:Itanium]`
`CPUArch`	Architecture du processeur	Architecture des processeurs 64 bits PowerPC et Power Architecture, à savoir : • ppc4 • POWER4 • POWER5 • POWER6 • POWER7 Par exemple : `[CPUArch:ppc4]`
`@<EnvVar_Name>`	Variable d'environnement d'un produit	Adhère aux règles de ce produit, par exemple : `[@TPAE_DB_SERVER:True]`

Exemple de fichier de configuration pour Windows utilisant des sections

Cet exemple utilise des sections pour classer par catégories les propriétés de prérequis pour une machine Windows, puis pour des machines exécutant des versions spécifiques de Windows.

```
#Properties for all Windows operating systems, that is, Windows XP and above
[OSType:Windows]
os.versionNumber=5.1+
network.pingSelf=True
network.pingLocalhost=True
network.availablePorts.Derby=4130
```

```
env.CIT.homeExists=True
env.classpath.derbyJAR=False
# Disk space properties
commonPath=10MB
installPath=200MB
tempPath=30MB

[OSType:Windows Vista]
os.servicePack=2+
```

Lorsque vous exécutez Prerequisite Scanner, le système analyse et recherche les différentes propriétés de prérequis selon le système d'exploitation et la version installée sur la machine.

Par exemple, tableau 7 présente les différentes sections contenant les propriétés de prérequis contrôlées basées sur l'exemple.

Tableau 7. Sections analysées d'un fichier de configuration pour Windows

Plateforme ou système d'exploitation	Sections avec des propriétés de prérequis
Machine avec Windows XP et ultérieur	`[OSType:Windows]`
Machine uniquement avec Windows Vista	`[OSType:Windows]` `[OSType:Windows Vista]`

Exemple de fichier de configuration pour UNIX utilisant des sections

Cet exemple contient les propriétés de prérequis de toutes les plateformes, des plateformes individuelles et des versions de systèmes d'exploitation pour un produit spécifique.

```
# Properties common to all UNIX platforms
[OSType:UNIX]
os.space.var=[dir:root=/var/ibm/common/acsi,unit:MB]1.0
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:MB]200
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
os.space.tmp=30MB
env.classpath.derbyJAR=False
network.pingSelf=True

# Properties common to all Linux platforms
[OSType:LINUX]
os.shell.default=bash
os.SELinux=[source:Command]Disabled
os.package.rpm=rpm

# Properties common to Linux platforms with the ppc64 CPU architecture
[OSType:LINUX][CPUArch:ppc64]
os.package.vacpp.rte=vacpp.rte-9.0.0-5+

# Properties common to all RedHat OS
[OSType:RedHat]
env.classpath.derbyJAR=False

# Properties common to all versions of Red Hat Enterprise
# Linux Server OS
[OSType: RedHatEnterpriseLinuxServer]
network.pingLocalhost=True

# Properties common to all Red Hat Enterprise Linux Server
# OS Version 6.x(6.1,6.2...)
[OSType: RedHatEnterpriseLinuxServer6.*]
os.package.compat-libstdc++-33=compat_libstdc++_33-3.2.3-68
```

```
[OSType:RedHatEnterpriseLinuxServer5.*]
os.package.compat-libstdc++-33=compat_libstdc++_33

# Properties common to all Red Hat Enterprise Linux Server
# Version 4.x(6.1,6.2...) OS and for Itanium  family CPU
[OSType:RedHatEnterpriseLinuxServer4.*][CPU:Itanium]
os.package.ia32el=ia32el-1.1-20

# Properties common to all Red Hat Enterprise Linux Server
# Version 4.x(6.1,6.2...) OS and for a 64-bit OS architecture
[OSType:RedHatEnterpriseLinuxServer4.*][OSArch:64-bit]
os.package.libgcc=libgcc-3.4.3-9

# Properties specific to RedHatEnterpriseLinuxServer5.2 OS
[OSType:RedHatEnterpriseLinuxServer5.2]
network.availablePorts.Derby=4130

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
[OSType:SUSELinuxEnterpriseServer11][OSArch:64-bit]
os.package.libstdc++33-32bit=libstdc++33_32bit-3.3.3-11.9

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
# and if the environment variable TPAE_DB_Server is set to 'True'
[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]
os.package.libstdc++31-32bit=libstdc++31_32bit

# Properties specific to a 64 bit SUSE Linux Enterprise Server 11 OS
# and if the environment variables TPAE_DB_Server and TPAE_DIR_Server
# are set to 'True'
[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]
[@TPAE_DIR_Server:True]
os.package.libstdc++34-32bit=libstdc++34_32bit

# Properties common to all AIX platforms
os.ulimit=[type:filesizelimit]unlimited
os.ulimit=[type:filedescriptorlimit]8192+,unlimited
os.FreePagingSpace=4GB+

# Properties specific to AIX 5.3.0.0 and
# if the environment variables TPAE_DB_FEATURE or TPAE_DIR_FEATURE
# are set to 'True'
[OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]
os.lib.xlC.aix50.rte=xlC.aix50.rte.9.0.0.8+
```

Lorsque vous exécutez Prerequisite Scanner, le système analyse et recherche les différentes propriétés de prérequis selon le système d'exploitation et la version installée sur la machine.

Par exemple, tableau 7, à la page 20 présente les différentes sections contenant les propriétés de prérequis contrôlées basées sur l'exemple.

Tableau 8. Sections analysées d'un fichier de configuration pour UNIX

Systèmes d'exploitation et versions	Sections avec des propriétés de conditions prérequises
Machine avec SUSE Linux Enterprise Server 11 64 bits	[OSType:UNIX] [OSType:LINUX] [OSType:LINUX][CPUArch:ppc64] [OSType:SUSE Linux Enterprise Server 11] [OSArch:64-bit]
Machine avec Red Hat Enterprise Linux Server 6.3	[OSType:UNIX] [OSType:LINUX] [OSType:RedHat] [OSType:RedHatEnterpriseLinuxServer] [OSType:RedHatEnterpriseLinuxServer6.*]

Tableau 8. Sections analysées d'un fichier de configuration pour UNIX (suite)

Systèmes d'exploitation et versions	Sections avec des propriétés de conditions prérequises

Machine avec SUSE Linux Enterprise Server 11 et la variable d'environnement `@TPAE_DB_Server` paramétrée sur `true`	`[OSType:UNIX]`
	`[OSType:LINUX]`
	`[OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]`

Machine avec AIX 5.3.0.0 et les variables d'environnement `@TPAE_DB_FEATURE` ou `@TPAE_DIR_FEATURE` définies sur `True`	`[OSType:UNIX]`
	`[OSType:AIX]`
	`[OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]`

collecteurs Prerequisite Scanner

Les collecteurs IBM Prerequisite Scanner recueille des données réelles sur l'environnement actuel, en fonction de l'ensemble des propriétés de prérequis des produits à installer. Les collecteurs reçoivent les données par le biais du code natif. Les données peuvent être des données courantes, telles que la vitesse du processeur et la RAM, les données logicielles installées, les données du système d'exploitation, les données utilisateur, le réseau et les données de connectivité. Les collectionneurs sont également extensibles. Par conséquent, vous pouvez créer des collecteurs personnalisés pour obtenir les valeurs réelles des propriétés de prérequis personnalisées.

Prerequisite Scanner utilise des collecteurs dans les langues suivantes en fonction de votre plateforme :

- Windows : VBScript avec l'extension `.vbs`
- UNIX: Shell avec ou sans l'extension `.sh`

Remarque : Vous ne pouvez pas exécuter les scripts UNIX sur les systèmes Windows, même si vous avez installé des environnements de type UNIX sur les machines Windows, par exemple, Cygwin.

Collecteurs pour systèmes Windows

Les collecteurs VBScript pour systèmes Windows sont exécutés dans l'environnement Windows Script Host. Ils utilisent le modèle COM pour accéder aux éléments de l'environnement Windows, par exemple, `FileSystemObject` et `TextStream`.

Prerequisite Scanner exécute les collecteurs VBScript pour obtenir les valeurs réelles des propriétés de prérequis pour l'environnement Windows. Chaque collecteur peut obtenir des données pour une ou plusieurs propriétés de prérequis.

Pour chaque propriété de prérequis dans un collecteur VBScript, le collecteur écrit le nom de la propriété de prérequis et sa valeur actuelle en tant que sortie standard. Prerequisite Scanner écrit cette sortie standard vers un fichier texte temporaire, qui est, `localhost_hw.txt`.

Vous pouvez créer des collecteurs VBScript communs personnalisés afin de collecter des données pour les propriétés de prérequis qui s'appliquent à n'importe quel produit et n'importe quelle version du produit. Vous pouvez également créer des collecteurs personnalisés spécifiques à un produit afin de collecter des données qui s'appliquent à un produit spécifique et à une version du produit.

Lorsque vous exécutez Prerequisite Scanner, il exécute les collecteurs dans l'ordre suivant: collecteurs VBScript prédéfinis; les collecteurs VBScript communs et

personnalisés dans le *ips_root*/`répertoirelib`; et les collecteurs VBScript spécifiques à un produit commun en recherchant le *product_code*[_*<version>*]`.fichier` vbs dans le *ips_root*/`répertoire` Windows.

Par exemple, le `env.tcrhome.vbs` fichier est un collecteur personnalisé qui vérifie le répertoire de base variable d'environnement pour Tivoli Common Reporting. Il est enregistré dans le *ips_root*/`répertoire` lib.

Les collecteurs VBScript doivent se conformer aux règles suivantes:

- convention d'attribution de noms pour les fichiers collecteur VBScript communs et personnalisés

  Il contient une propriété de prérequis devant être disponible pour n'importe quel produit et version de produit, donc, tous les fichiers de configuration:

  prefix_identifier.]`property_name.vbs`

  Où :

  – *prefix_identifier* est le préfixe identificateur pour une catégorie prédéfinie de propriétés de prérequis comme décrit dans tableau 3, à la page 4. Ce préfixe identificateur est exigé par certaines des catégorie prédéfinies, par exemple, `env`.

  – *property_name* est le nom de propriété de prérequis, par exemple, `tcrhome`.

  Enregistre ce type de collecteur VBScript dans le *ips_root*/`répertoire` lib.

- Convention d'attribution de noms pour les fichiers collecteur VBScript spécifiques à un produit personnalisé

  Il contient des propriétés devant être disponibles à un produit spécifique et à des versions de produit, c'est-à-dire, un fichier de configuration:

  product_code[_*<version>*]`.vbs`

  Où :

  – *product_code*

    Il s'agit de la variable représentant un code produit sur les systèmes Windows ou UNIX. Les codes produit identifient le produit, une plateforme individuelle, telle que Windows, AIX, HP-UX, Linux et Solaris, et éventuellement la version du système d'exploitation prise en charge par ce produit. Ils sont stockés dans le fichier `codename.cfg`. Tout produit prenant en charge plusieurs plateformes comprend plusieurs codes produit, chacun identifiant un produit, une plateforme et la version du système d'exploitation, selon les besoins.

  – *<version>* est le code à 8 chiffre pour représenter la version, la publication, la modification, et le niveau, avec deux chiffre pour chaque partie du code; par exemple, 7.3.21 est 07032100.

  Enregistre ce type de collecteur VBScript dans le *ips_root*/`répertoire` Windows.

- La sortie standard pour chaque propriété de prérequis se présente comme suit:

  `WScript.Echo "`*property_name*`=" & `*<var_for_value>*

  – *property_name* qui représente la propriété de prérequis comme écrit dans le fichier de configuration, par exemple,

    `env.tcrhome.`

  – *var_for_value*, qui est, la variable VBScript pour la valeur réelle obtenue par le collecteur pour la propriété de prérequis.

Par exemple, la sortie standard suivante écrit la propriété de prérequis pour la Tivoli Common Reporting la variable d'environnement de base et sa valeur réelle:

```
WScript.Echo "env.tcrhome=" & tcr_home
```

Collecteurs pour les systèmes UNIX

Les collecteurs pour les systèmes UNIX sont exécutés dans un environnement d'hébergement pertinent Shell pour AIX, HP-UX, Linux ou Solaris. Ils utilisent les commandes et les options spécifiques à cette plateforme pour accéder aux éléments de l'environnement d'hébergement.

Chaque collecteur UNIX reçoit les données d'une propriété de prérequis ou une propriété de prérequis avec des sous-types prédéfinis. Le collecteur écrit le résultat de la vérification de la propriété de prérequis en tant que sortie standard. Prerequisite Scanner écrit cette sortie standard sur un fichier texte temporaire.

Vous pouvez créer des collecteurs UNIX personnalisés pour recueillir les données des propriétés de prérequis personnalisées. Chaque collecteur, prédéfini ou personnalisé, est appelé dans le fichier *ips_root*/`UNIX_Linux/packageTest.sh`.

Lorsque vous exécutez Prerequisite Scanner, il exécute les collecteurs dans l'ordre suivant : les collecteurs prédéfinis avec `_plug` dans le nom du fichier du répertoire *ips_root*/`lib`; les collecteurs prédéfinis dans le répertoire *ips_root*/`UNIX_Linux` et les collecteurs UNIX personnalisés dans le répertoire *ips_root*/`UNIX_Linux`.

Par exemple, le fichier `installedSoftware.TCR.version` désigne un collecteur personnalisé qui obtient la version Tivoli Common Reporting installée sur la machine. Il est stocké dans le répertoire *ips_root*/`UNIX_Linux`.

Les UNIX doivent respecter les règles suivantes :

- La convention de dénomination du fichier de collecteur UNIX personnalisé sans extension de fichier :

  [*prefix_identifier*.]*property_name*

  où :

  – *prefix_identifier* désigne un identificateur d'une catégorie prédéfinie de propriétés de prérequis tel que décrit dans tableau 3, à la page 4. Cet identificateur à préfixe est obligatoire dans certaines des catégories prédéfinies, par exemple, `installedSoftware`.

  – *property_name* désigne le nom de la propriété de prérequis, par exemple, `TCR.version`.

  Stockez le collecteur dans le répertoire *ips_root*/`UNIX_Linux`. Assurez-vous qu'il ne s'agit pas d'une extension de fichier.

- La sortie standard d'une propriété de prérequis renvoie la valeur réelle de la propriété de prérequis s'il s'agit d'un entier ou d'une chaîne, par exemple, la version logicielle ou la quantité d'espace disque disponible pour l'installation d'un système de fichiers. Sinon, elle peut renvoyer `"Unavailable"`.

```
echo "True"|"False"
'If the scan checks for the existence of the prerequisite
'property
echo $res
'If the scan checks returns the value, for example, product version,
'of the prerequisite property
echo "Unavailable"
'If the scan returns no value for the prerequisite property
echo "Available"
'If the scan returns a valid check for the prerequisite property
```

- Effectuez un appel codé et exécutez le collecteur dans le script *ips_root*/UNIX_Linux/packageTest.sh.

```
res=`echo $line | grep installedSoftware.TCR.version`
si [ $res ]; ensuite
ExpValue=`echo $res | cut -d "=" -f2`

 echo "\`wrlTrace "Starting" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Executing" "installedSoftware.TCR.version"\`" >>/tmp/prs.check
 echo "\`wrlDebug "Starting" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check


 echo "ss=\`./installedSoftware.TCR.version\`" >>/tmp/prs.check
 echo "\`wrlTrace "Finished" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "echo \"os.userLimits=\$ss\"" >>/tmp/prs.check
 echo "\`wrlDebug "Finished" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "OutPutValueIs" \$ss\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Done" "installedSoftware.TCR.version"\`"  >>/tmp/prs.check
fi
```

Evaluateurs Prerequisite Scanner

Les évaluateurs IBM Prerequisite Scanner sont des scripts qui comparent les données réelles provenant des collecteurs et les données attendues pour les mêmes propriétés figurant dans les fichiers de configuration. Les évaluations peuvent être les suivantes : spécifiques à la plateforme, basées sur les opérateurs simples comme inférieur à, égal à ou supérieur à, et basées sur le fait qu'une propriété est installée, présente ou activée. Elles peuvent également vérifier que les ports sont en cours d'utilisation ou disponibles ainsi que le statut de connectivité de la machine. Vous pouvez créer ou modifier les évaluateurs.

Prerequisite Scanner utilise les évaluateurs dans les langues suivantes, en fonction de votre plateforme :

- Windows : VBScript avec extension .vbs
- UNIX : interpréteur de commandes avec extension .sh

Remarque : Vous ne pouvez pas exécuter les scripts UNIX sur les systèmes Windows même si vous avez installé un environnement de type UNIX sur les machines Windows, par exemple Cygwin.

Stockez les évaluateurs dans *ips_root*/*OS*, où *OS* correspond au nom du système d'exploitation, par exemple Windows ou UNIX_Linux.

Les fichiers de l'évaluateur doivent être conformes aux règles suivantes :

- Convention de dénomination pour le nom de fichier :

  [*prefix_identifier*.]*property_name*[.*suffix_identifier*]_compare.vbs|sh

  où :

  – *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis, comme indiqué dans tableau 3, à la page 4. Cet identificateur à préfixe est obligatoire pour certaines des catégories prédéfinies.
  – *property_name* est le nom de la propriété de prérequis.
  – *suffix_identifier* est un identificateur facultatif pour un sous-type de propriétés de prérequis, comme indiqué dans tableau 4, à la page 7.

- Transmettez éventuellement deux paramètres d'entrée au script pour les évaluations complexes :
  - *expected_value*, c'est-à-dire la valeur attendue pour la propriété de prérequis définie dans le fichier de configuration.
  - *actual_value*, c'est-à-dire la valeur réelle que le collecteur reconnaît pour la propriété de prérequis de la machine.
- La sortie standard se présente comme suit :
  - `"PASS"` lorsque la valeur attendue pour la propriété de prérequis est supérieure ou égale à la valeur réelle de la propriété de prérequis.
  - `"FAIL"` lorsque la valeur attendue pour la propriété de prérequis est différente de la valeur réelle de la propriété de prérequis.

Formats de sortie

IBM Prerequisite Scanner génère une sortie pour l'écran et les formats de fichier lisibles suivants : sortie vers l'interface de ligne de commande, fichiers journaux de débogage et de trace, et fichiers texte et XML pour les résultats.

Prerequisite Scanner enregistre les résultats de l'analyse et les fichiers journaux sous le répertoire *ips_output_dir*. Vous pouvez paramétrer ce répertoire à l'aide du paramètre d'entrée **`outputDir`** lorsque vous exécutez le scanner. Si vous ne définissez pas ce paramètre, l'emplacement de sortie par défaut est *ips_root*.

Remarque : Prerequisite Scanner crée des fichiers temporaires au cours de son exécution, mais ces fichiers sont supprimés avant que le scanner termine son exécution. Ces fichiers temporaires se trouvent dans le sous-répertoire *ips_output_dir*`/temp`. Le scanner supprime également le sous-répertoire *ips_output_dir*`/temp` à moins que ce dernier ne contienne les fichiers de débogage et de trace générés uniquement sous les systèmes UNIX.

Vous pouvez également utiliser le paramètre pour spécifier un emplacement si vous décidez d'exécuter Prerequisite Scanner à partir d'un CD, d'un DVD ou d'une unité réseau en lecture seule.

Important : Si le répertoire de sortie n'existe pas, Prerequisite Scanner le crée. Vous devez disposer des droits en écriture pour créer ou écrire dans le répertoire de sortie dans lequel Prerequisite Scanner enregistre les fichiers.

Sortie de l'interface de ligne de commande

Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **`detail`**, Prerequisite Scanner affiche les résultats détaillés de l'analyse dans l'interface de ligne de commande. Les résultats détaillés contiennent les éléments suivants :

- Version de Prerequisite Scanner
- Version du système d'exploitation sur lequel le scanner a été exécuté
- Type d'analyse et scénario
  - Analyses prérequises : `Scénario : Analyse prérequise`
- Nom des produits ou des composants pour lesquels les contrôles ou les diagnostics d'intégrité prérequis ont été effectués
- Pour chaque propriété de prérequis : nom de la propriété contrôlée, résultat PASS ou FAIL, valeur réelle et valeur attendue

- Pour tous les composants : nom de la propriété générale contrôlée, résultat PASS ou FAIL, valeur réelle et valeur attendue
- Résultat PASS ou FAIL général, avec toute défaillance d'un contrôle individuel entraînant la défaillance de l'analyse globale

```
Command Prompt

C:\prs\precheck_windows_20110927>prereq_checker.bat DMO detail

IBM Prerequisite Scanner
    Version   : 1.1.1.8
    Build     : 20110927
    OS Name   : Microsoft Windows XP Professional Service Pack 3
    User Name: <User Name>

Machine Info
   Machine name : <Machine name>
   Serial Number: <Serial number>
   OS Serial    : <OS serial number>

DMO - Prerequisite Scanner Demo [version 01000000]:

Property                              Result  Found            Expected
========                              ======  =====            ========
OS Version                            PASS    Microsoft Win... regex(Windows...
Memory                                PASS    645MB            128MB
Disk#1 (C:\ibm\ITM)                   PASS    1.38GB           1.00GB
os.versionNumber                      PASS    5.1.2600         5.1.*
os.servicePack                        PASS    3.0              2+
os.architecture                       PASS    32-bit           32-bit
os.totalPhysicalMemory                PASS    3.00GB           2.00GB
os.is8dot3FileFormatEnabled           PASS    True             True
os.isServiceRunning.terminalServices  PASS    True             True
os.isServiceRunning.remoteRegistry    FAIL    True             False
os.isServiceRunning.DNSClient         PASS    True             True
user.isAdmin                          PASS    True             True
network.availablePorts.DB             PASS    135,445,523,1... 60000-60005
network.availablePorts.WAS            PASS    135,445,523,1... 8080
network.availablePorts.FTP            PASS    135,445,523,1... 21
network.netBIOSEnabled                PASS    True             True
network.pingSelf                      PASS    True             True
network.DHCPEnabled                   FAIL    True             False
cygwinVersion                         FAIL    0.0              1.5+

ALL COMPONENTS :
Property                              Result  Found            Expected
========                              ======  =====            ========
Memory                                PASS    645MB            128MB
C:                                    PASS    1.38GB           1.00GB

Prereq Scanner Overall Result: FAIL

Details also available in C:\prs\precheck_windows_20110927\result.txt

C:\prs\precheck_windows_20110927>_
```

Figure 1. Sortie vers l'interface de ligne de commande sous les systèmes Windows

Si vous ne définissez pas le paramètre **detail**, le scanner affiche uniquement le résultat PASS ou FAIL à l'écran.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
[root@aclinux15 20110927-0849]# ./prereq_checker.sh DMO
IBM Prerequisite Scanner
     Version: 1.1.1.8
     Build  : 20110927
     OS Name: Linux

 Machine Info
 Machine Name : <Machine name>
 Serial Number: <Serial number>

TPS detected : Red Hat Enterprise Linux Server release 5.5 {32-bit}
Using the DMO config file
Using config file - /root/prs/20110927-0849/UNIX_Linux/DMO_0750000.cfg for DMO
FAIL
[root@aclinux15 20110927-0849]#
```

Figure 2. Sortie vers l'interface de ligne de commande sur les systèmes UNIX

Prerequisite Scanner génère des codes de retour en fonction des résultats de l'analyse et du fait qu'il faille éventuellement quitter le système en raison d'erreurs. Ces codes de retour sont écrits dans les fichiers journaux. Par exemple, si Prerequisite Scanner ne parvient pas à effectuer l'analyse car il ne peut pas lire le fichier de configuration, il génère le code de retour de 2.

Cumul des propriétés de prérequis associés à la mémoire et à l'espace disque

Vous pouvez exécuter Prerequisite Scanner pour vérifier simultanément les prérequis d'un ou de plusieurs produits ou composants lorsque vous spécifiez plusieurs codes produits comme paramètres d'entrée. Prerequisite Scanner cumule les résultats des contrôles de prérequis associés à la mémoire et à l'espace disque dans les sections cumulées suivantes de la sortie si les propriétés de prérequis associés ont été spécifiées dans l'un des fichiers de configuration :

- Sur les systèmes UNIX, dans la section `TOTAL ALL SPECIFIED COMPONENTS`
- Sur les systèmes Windows, dans la section `ALL COMPONENTS`

Les propriétés de prérequis générales se présentent comme suit :

- Taille totale de mémoire physique actuellement disponible dans l'environnement cible, à savoir :

  `Memory`
- Espace disque des systèmes de fichiers pour les propriétés de prérequis suivantes :

Plateforme	Propriétés de prérequis
UNIX et Linux	• `os.space.home` • `os.space.opt` • `os.space.tmp` • `os.space.usr` • `os.space.var`
Windows	`Disk`

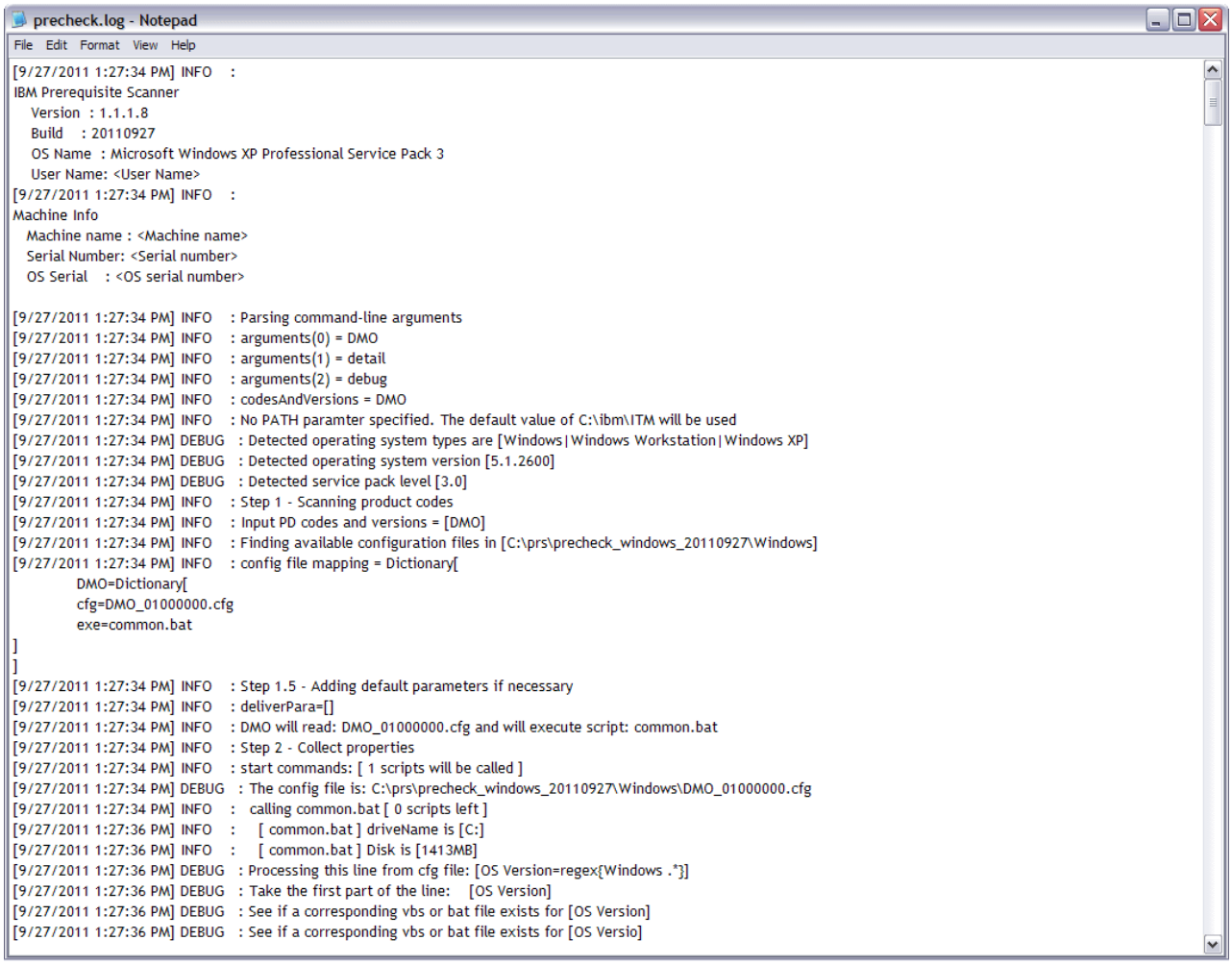
Remarque : Si `opt`, `usr` et `var` n'ont pas été définis en tant que systèmes de fichiers sur l'ordinateur cible, Prerequisite Scanner n'affiche pas les valeurs attendues et les résultats retournés pour ces propriétés de prérequis dans la section cumulée.

Prerequisite Scanner n'affiche pas la section cumulée si ni les propriétés de prérequis de la mémoire, ni celles de l'espace disque n'existent dans les fichiers de configuration.

Prerequisite Scanner gère la comparaison et l'affichage des valeurs de l'espace disque dans la section cumulée des résultats de l'analyse différemment de la section principale. Voir «Unités de mesure en sortie», à la page 34.

Sortie du fichier journal de débogage sur les systèmes Windows

Prerequisite Scanner affiche les informations de traitement, les messages d'avertissement et d'erreur ainsi que les résultats d'analyse dans le fichier *ips_output_dir*/`precheck.log`. Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **`debug`**, Prerequisite Scanner affiche les messages de débogage supplémentaires dans ce fichier.




```
[9/27/2011 1:27:34 PM] INFO   :
IBM Prerequisite Scanner
   Version  : 1.1.1.8
   Build    : 20110927
   OS Name  : Microsoft Windows XP Professional Service Pack 3
   User Name: <User Name>
[9/27/2011 1:27:34 PM] INFO   :
Machine Info
  Machine name : <Machine name>
  Serial Number: <Serial number>
  OS Serial    : <OS serial number>

[9/27/2011 1:27:34 PM] INFO   : Parsing command-line arguments
[9/27/2011 1:27:34 PM] INFO   : arguments(0) = DMO
[9/27/2011 1:27:34 PM] INFO   : arguments(1) = detail
[9/27/2011 1:27:34 PM] INFO   : arguments(2) = debug
[9/27/2011 1:27:34 PM] INFO   : codesAndVersions = DMO
[9/27/2011 1:27:34 PM] INFO   : No PATH paramter specified. The default value of C:\ibm\ITM will be used
[9/27/2011 1:27:34 PM] DEBUG  : Detected operating system types are [Windows|Windows Workstation|Windows XP]
[9/27/2011 1:27:34 PM] DEBUG  : Detected operating system version [5.1.2600]
[9/27/2011 1:27:34 PM] DEBUG  : Detected service pack level [3.0]
[9/27/2011 1:27:34 PM] INFO   : Step 1 - Scanning product codes
[9/27/2011 1:27:34 PM] INFO   : Input PD codes and versions = [DMO]
[9/27/2011 1:27:34 PM] INFO   : Finding available configuration files in [C:\prs\precheck_windows_20110927\Windows]
[9/27/2011 1:27:34 PM] INFO   : config file mapping = Dictionary[
        DMO=Dictionary[
        cfg=DMO_01000000.cfg
        exe=common.bat
]
]
[9/27/2011 1:27:34 PM] INFO   : Step 1.5 - Adding default parameters if necessary
[9/27/2011 1:27:34 PM] INFO   : deliverPara=[]
[9/27/2011 1:27:34 PM] INFO   : DMO will read: DMO_01000000.cfg and will execute script: common.bat
[9/27/2011 1:27:34 PM] INFO   : Step 2 - Collect properties
[9/27/2011 1:27:34 PM] INFO   : start commands: [ 1 scripts will be called ]
[9/27/2011 1:27:34 PM] DEBUG  : The config file is: C:\prs\precheck_windows_20110927\Windows\DMO_01000000.cfg
[9/27/2011 1:27:34 PM] INFO   :   calling common.bat [ 0 scripts left ]
[9/27/2011 1:27:36 PM] INFO   :     [ common.bat ] driveName is [C:]
[9/27/2011 1:27:36 PM] INFO   :     [ common.bat ] Disk is [1413MB]
[9/27/2011 1:27:36 PM] DEBUG  : Processing this line from cfg file: [OS Version=regex{Windows .*}]
[9/27/2011 1:27:36 PM] DEBUG  : Take the first part of the line:    [OS Version]
[9/27/2011 1:27:36 PM] DEBUG  : See if a corresponding vbs or bat file exists for [OS Version]
[9/27/2011 1:27:36 PM] DEBUG  : See if a corresponding vbs or bat file exists for [OS Versio]
```

Figure 3. fichier precheck.log

Sortie du fichier journal de débogage et de trace sur les systèmes UNIX

Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **debug**, Prerequisite Scanner affiche les informations de traitement détaillées, les messages d'avertissement et d'erreur ainsi que les résultats d'analyse dans le fichier *ips_output_dir*/`temp/prs.debug`.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
     Build  : 20110927
     OS Name: Linux
[2011.09.27 10.12.15 ] [main()] - Entered
[2011.09.27 10.12.15 ]  ==== Step 1: Detecting OS...
[2011.09.27 10.12.15 ]  OS Detected: Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  product_version: DMO
[2011.09.27 10.12.15 ] [AutoOsDetection()] - Entered
[2011.09.27 10.12.15 ]  [Param] ProductInfo:DMO
[2011.09.27 10.12.15 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  [Param] CPU Arch:Kernel=i686
[2011.09.27 10.12.15 ]  Finding product code in product.cfg
[2011.09.27 10.12.15 ]  product code found :
[2011.09.27 10.12.15 ]  Found DMO code in product.cfg
[2011.09.27 10.12.15 ]  Finding OS Arch and CPU Type
[2011.09.27 10.12.15 ]  Found OS Arch = 32-bit, CPU Type=
[2011.09.27 10.12.15 ]  Calling config_parser.sh...
[2011.09.27 10.12.15 ] [config_parser.sh] - Entered
[2011.09.27 10.12.15 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  [Param] ProductCode:DMO
[2011.09.27 10.12.15 ]  [Param] OSArch:Arch=32-bit
[2011.09.27 10.12.16 ]  [Param] CPUArch:CPU=
[2011.09.27 10.12.16 ]  [Param] Version:version=
[2011.09.27 10.12.16 ]  [Param] XXX:Kernel=i686
[2011.09.27 10.12.16 ]  Forming parse array...
[2011.09.27 10.12.16 ] [Form_Parse_String] - Entered
[2011.09.27 10.12.16 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.16 ]  [Param] ProductCode:DMO
[2011.09.27 10.12.16 ]  [Param] OSArch:Arch=32-bit
[2011.09.27 10.12.16 ]  [Param] CPU:CPU=
[2011.09.27 10.12.16 ]  [Param] CPUArch:Kernel=i686
[2011.09.27 10.12.16 ]  Form_Parse_String - ParseArray: [OSType:UNIX][OSType:LINUX][OSType:RedHat][OSType:RedHatEnterpriseLinuxServer][OS
Type:RedHatEnterpriseLinuxServer5.*][OSType:RedHatEnterpriseLinuxServer5.5][OSArch:32-bit][CPUArch:i686]
[2011.09.27 10.12.16 ] [Form_Parse_String] - Exit
[2011.09.27 10.12.16 ]  Reading config file and parsing using parse array...
[2011.09.27 10.12.16 ] [Read_configFile()] - Entered
[2011.09.27 10.12.16 ]  [Param] ConfigFile:/root/prs/20110927-0849/UNIX_Linux/DMO_0750000.cfg-Master
[2011.09.27 10.12.16 ]  [Param] Product:DMO
[2011.09.27 10.12.17 ]  Writing DBType=Oracle to DMO_0750000.cfg
[2011.09.27 10.12.21 ]  Found Env Var - TPAE_DB_Server
```

Figure 4. fichier prs.debug sur les systèmes UNIX

Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **trace**, Prerequisite Scanner affiche les informations de trace dans le fichier *ips_output_dir*/temp/prs.trc.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
     Build  : 20110927
     OS Name: Linux
[2011.09.27 10.19.58 ] [main()] - Entered:
[2011.09.27 10.19.58 ] [AutoOsDetection()] - Entered:
[2011.09.27 10.19.58 ] [config_parser.sh] - Entered:
[2011.09.27 10.19.59 ] [Form_Parse_String] - Entered:
[2011.09.27 10.19.59 ] [Form_Parse_String] - Exit:
[2011.09.27 10.19.59 ] [Read_configFile()] - Entered:
[2011.09.27 10.20.05 ] [Read_configFile()] - Exit:
[2011.09.27 10.20.05 ] [config_parser.sh] - Exit:
[2011.09.27 10.20.05 ] [AutoOsDetection()] - Exit:
[2011.09.27 10.20.05 ] [packageTest.sh] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Exit:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Exit:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.26 ] [NFScheck()] - Exit:
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DB2_Version
[2011.09.27 10.20.26 ] Executing: DB2_Version.sh
[2011.09.27 10.20.26 ] Finished: DB2_Version.sh
[2011.09.27 10.20.26 ] Done    : DB2_Version
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DB2_Version
[2011.09.27 10.20.26 ] Executing: DB2_Version.sh
[2011.09.27 10.20.26 ] Finished: DB2_Version.sh
[2011.09.27 10.20.26 ] Done    : DB2_Version
[2011.09.27 10.20.26 ] Starting: DBType
```

Figure 5. fichier prs.trc sur les systèmes UNIX

Sortie du fichier texte

Prerequisite Scanner affiche les résultats d'analyse détaillés dans le fichier *ips_output_dir*/`result.txt`. Il enregistre les résultats dans le fichier texte, que vous définissiez ou non le paramètre **`detail`**.

```
result.txt - Notepad
File  Edit  Format  View  Help
IBM Prerequisite Scanner
   Version  : 1.1.1.8
   Build    : 20110927
   OS Name  : Microsoft Windows XP Professional Service Pack 3
   User Name: <User Name>
Machine Info
  Machine name : <Machine name>
  Serial Number: <Serial number>
  OS Serial    : <OS serial number>

DMO - Prerequisite Scanner Demo [version 01000000]:

Property                          Result  Found                                          Expected
========                          ======  =====                                          ========
OS Version                        PASS    Microsoft Windows XP Professional Service Pack 3  regex{Windows .*}
Memory                            PASS    645MB                                          128MB
Disk#1 (C:\ibm\ITM)               PASS    1.38GB                                         1.00GB
os.versionNumber                  PASS    5.1.2600                                       5.1.*
os.servicePack                    PASS    3.0                                            2+
os.architecture                   PASS    32-bit                                         32-bit
os.totalPhysicalMemory            PASS    3.00GB                                         2.00GB
os.is8dot3FileFormatEnabled       PASS    True                                           True
os.isServiceRunning.terminalServices  PASS  True                                         True
os.isServiceRunning.remoteRegistry  FAIL  True                                           False
os.isServiceRunning.DNSClient     PASS    True                                           True
user.isAdmin                      PASS    True                                           True
network.availablePorts.DB         PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...               60000-60005
network.availablePorts.WAS        PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...               8080
network.availablePorts.FTP        PASS    135,445,523,1035,1067,1099,1527,2967,3389,5157,
                                          16310,16311,16312,16313,16315...               21
network.netBIOSEnabled            PASS    True                                           True
network.pingSelf                  PASS    True                                           True
network.DHCPEnabled               FAIL    True                                           False
cygwinVersion                     FAIL    0.0                                            1.5+

ALL COMPONENTS :
Property                          Result  Found                                          Expected
========                          ======  =====                                          ========
Memory                            PASS    645MB                                          128MB
C:                                PASS    1415MB                                         1024MB
Prereq Scanner Overall Result: FAIL
```

Figure 6. fichier result.txt sur les systèmes Windows

```
[root@aclinux15 20110927-0849]# cat result.txt
IBM Prerequisite Scanner
     Version: 1.1.1.8
     Build  : 20110927
     OS Name: Linux

 Machine Info
 Machine Name : <Machine name>
 Serial Number: <Serial number>

DMO - Prerequisite Scanner Demo [0750000]:
Evaluation                              PASS/FAIL       Result                                      Expected Result
DBType                                  FAIL            Unknown                                     Oracle
DBType                                  FAIL            Unknown                                     DB2
DBType                                  FAIL            Unknown                                     regex{.*Oracle.*}
DBType                                  FAIL            Unknown                                     regex{.*DB2.*}
DBTypeDetails                           FAIL            Unknown                                     oracle
DBTypeDetails                           FAIL            Unknown                                     DB2
DBTypeDetails                           FAIL            Unknown                                     regex{.*Oracle.*}
DBTypeDetails                           FAIL            Unknown                                     regex{.*DB2.*}
OS Version                              PASS            "Red Hat Enterprise Linux Server release 5.5 (Tikanga)" "regex{R
ed Hat.*Tikanga.*}"
                                                                                                    regex{AIX.*}
                                                                                                    regex{Solaris.*}
os.lib.libstdc++                        PASS            /usr/lib/gcc/i386-redhat-linux/4.1.1/libstdc++.solibstdc++

os.lib.libgcc                           PASS            /usr/lib/gcc/i386-redhat-linux/3.4.6/libgcc_s.so[CheckPackage:Tr
ue]regex{libgcc.*}
os.lib.libXp                            PASS            /usr/lib/libXmu.so.6                        regex{libX.*}
os.space.var                            PASS            "38GB"                                       "[dir:root=/var/ibm/
common/acsi"
                                                                                                    unit:MB]1.0
os.space.usr                            PASS            "38GB"                                       "[dir:root=/usr/ibm/
common/acsi"
                                                                                                    unit:MB]200
os.space.tmp                            PASS            36GB                                        30MB
env.classpath.derbyJAR                  PASS            False                                       False
network.pingSelf                        PASS            True                                        True
env.classpath.derbyJAR                  PASS            False                                       False
```

Figure 7. fichier result.txt sur les systèmes UNIX

Sortie du fichier XML

Prerequisite Scanner affiche les résultats d'analyse détaillés dans le fichier *ips_output_dir*/`result.xml` lorsque vous définissez le paramètre d'entrée **`xmlResult`** facultatif. Vous pouvez l'utiliser pour indiquer à l'outil d'afficher les résultats dans le fichier de résultats XML en plus du fichier de résultats de texte brut. Il enregistre les résultats sur le fichier XML, que vous définissiez ou non le paramètre **`detail`**.

```
<PRSInfo>

<MachineInfo>
    <MachineName>my_machine_name</MachineName>
    <MachineSerialNumber>serial_number</MachineSerialNumber>
    <MachineOSSerial>os_serial_number</MachineOSSerial>
    <MachineOSName>Microsoft Windows XP Professional Service Pack 3</MachineOSName>
</MachineInfo>

<UserInfo>

<ProductInfo>
    <ProductElement>
        <ProductCode>DMO</ProductCode>
        <ProductName>Prerequisite Scanner Demo</ProductName>
        <ProductVersion>01000000</ProductVersion>
    </ProductElement>
</ProductInfo>

<DetailedResults>
    <DetailedProductResultsElement>
        <ProductCode>DMO</ProductCode>
        <ResultElement>
            <PropertyName>OS Version</PropertyName>
            <Result>FAIL</Result>
            <Found>Microsoft Windows XP Professional Service Pack 3</Found>
            <Expected>Windows 7 Ultimate</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>Memory</PropertyName>
            <Result>PASS</Result>
            <Found>960MB</Found>
            <Expected>128MB</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>Disk#1 (C:\ibm\ITM)</PropertyName>
            <Result>PASS</Result>
            <Found>22072MB</Found>
            <Expected>1GB</Expected>
        </ResultElement>
        <ResultElement>
            <PropertyName>os.versionNumber</PropertyName>
            <Result>FAIL</Result>
            <Found>5.1.2600</Found>
            <Expected>5.2.*</Expected>
        </ResultElement>
```


Figure 8. fichier result.XML sur les systèmes Windows

Les développeurs de logiciel peuvent utiliser le kit d'outils Prerequisite Scanner Java Developer pour analyser et lire le fichier XML.

Unités de mesure en sortie

Prerequisite Scanner gère la comparaison et l'affichage des valeurs de l'espace disque dans la section cumulée des résultats de l'analyse différemment de la section principale.

Dans la section principale des résultats d'analyse, Prerequisite Scanner gère la comparaison et l'affichage des valeurs de l'espace disque comme suit :

Plateforme	Propriétés de prérequis
UNIX et Linux	Si la valeur réelle est supérieure à 1 024 Mo, Prerequisite Scanner convertit et renvoie la valeur en Go ; sinon, il renvoie la valeur en Mo.

Plateforme	Propriétés de prérequis
Windows	Prerequisite Scanner utilise l'unité de la valeur attendue dans le fichier de configuration comme unité de comparaison et d'affichage. Il convertit la valeur réelle dans cette unité si nécessaire.

Dans la section cumulée des résultats d'analyse, Prerequisite Scanner gère la comparaison et l'affichage des valeurs de l'espace disque comme suit :

Plateforme	Propriétés de prérequis
UNIX et Linux	Si la valeur réelle est supérieure à 1 024 Mo, Prerequisite Scanner convertit et renvoie la valeur en Go ; sinon, il renvoie la valeur en Mo.
Windows	Prerequisite Scanner effectue la comparaison de l'espace disque en se basant sur des unités en Mo. Pour chaque fichier de configuration contenant la propriété de prérequis, `Disk`, Prerequisite Scanner convertit les valeurs réelles et attendues en Mo et effectue la comparaison. Pour l'affichage, si la valeur réelle cumulée est supérieure à 1 Go, Prerequisite Scanner renvoie la valeur réelle en Go avec une précision de 2 ; sinon, il renvoie la valeur en Mo.

Kit d'outils Java Developer de Prerequisite Scanner

Le kit d'outils Prerequisite Scanner Java Developer est un ensemble d'interfaces API vous permettant, en tant que développeur de logiciel, d'analyser à l'aide d'un programme et de lire le contenu du fichier XML de résultats pour répondre à vos besoins ; par exemple analyser les résultats du balayage à utiliser dans votre programme d'installation.

Le kit d'outils comprend les modules suivants :

- com.ibm.prs.common.exception

  Il contient la classe `PRSApiException` qui fournit des méthodes pour émettre des exceptions concernant l'interface API de la requête XML.
- com.ibm.prs.common.reports.api

  Il contient l'interface `PRSXmlResultReader` qui définit l'interface API de la requête XML pour le fichier de résultats XML.
- com.ibm.prs.common.reports.api.impl

  Il contient la classe `PRSXmlResultReaderImpl` qui implémente `PRSXmlResultReader`.

Prerequisite Scanner peut valider le formatage et la structure en fonction du fichier de schéma XML *ips_root*`/PRSResults.xsd`.

Javadoc est disponible pour le kit d'outils dans le répertoire *ips_root*`/api/javadoc`.

Fichier de schéma XML pour le fichier de résultats XML

Prerequisite Scanner fournit un fichier de schéma XML à partir duquel le fichier de résultats XML peut être validé.

Le fichier de schéma XML contient les éléments suivants sous forme de sections :

- `PRSInfo` pour gérer les détails de Prerequisite Scanner
- `MachineInfo` pour gérer les informations sur l'environnement cible dans lequel le scan est exécuté
- `UserInfo` pour gérer les informations sur l'utilisateur connecté exécutant le scan
- `ScenarioInfo` pour gérer les informations sur le type de scan et de scénario
- `ProductInfo` pour gérer les informations sur le produit ou composant et son fichier de configuration
- `DetailedResults` pour gérer les résultats de scan pour chaque ensemble de propriétés de conditions requises pour un produit ou composant regroupés par `DetailedProductResultsElement`
- `AggregateResults` pour gérer les résultats de scan agrégés pour l'espace disque et la mémoire
- `OverallResult` pour gérer la totalité du résultat PASS ou FAIL du scan

Le nom et l'emplacement du schéma XML est :*ips_root*`/PRSResults.xsd`

En tant que développeur ou déployeur, vous pouvez recourir à des méthodes de l'API du XML de requête pour valider le fichier XML de résultats. Le Javadoc est disponible pour le kit d'outils dans le répertoire *ips_root*`/api/javadoc`.

Processus de numérisation

Lorsque vous exécutez IBM Prerequisite Scanner, il effectue un ensemble de tâches à chaque étape du processus de numérisation. L'utilisateur ouvre une interface de ligne de commande et exécute le script Prerequisite Scanner avec l'ensemble de paramètres d'entrée comprenant un code produit.

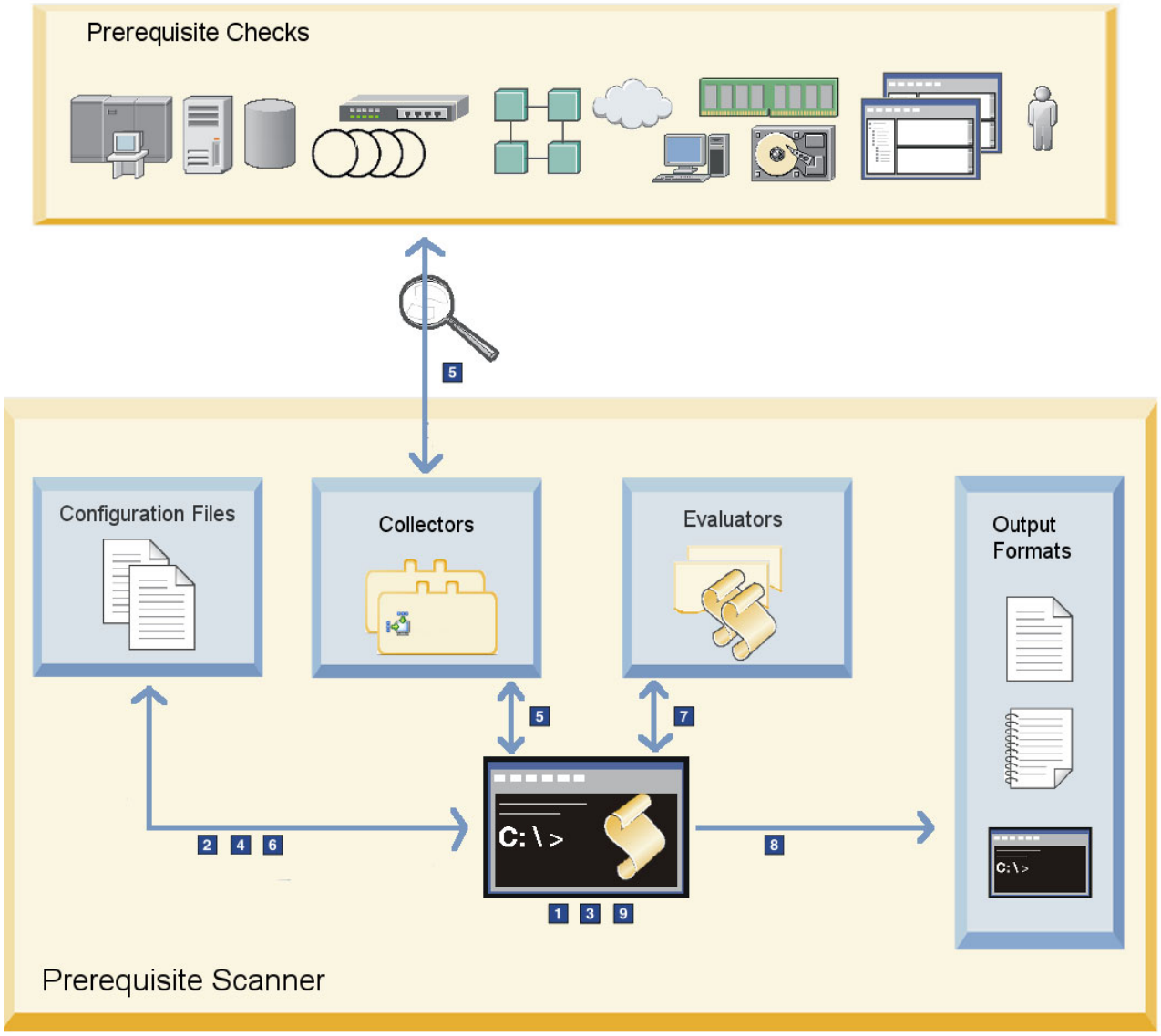


Figure 9. Prerequisite Scanner architecture et processus de numérisation

Le processus de numérisation de figure 9 se résume comme suit :

1. Prerequisite Scanner vérifie le format des paramètres d'entrée transmis au scanner.
2. Le scanner détermine si le code produit transmis comme l'un des paramètres d'entrée est un code produit valide du fichier `codename.cfg`.
3. Le scanner recherche le fichier de configuration associé au code produit. Si le paramètre de version de produit facultatif n'a pas été transmis, le scanner utilise la toute dernière version du fichier de configuration qu'il trouve dans le répertoire *ips_root*/`Windows|UNIX_Linux`.
4. Le scanner vérifie si le système d'exploitation actuel de la machine est un système d'exploitation pris en charge. Le scanner vérifie le système d'exploitation actuel en fonction du système d'exploitation pris en charge

attendu répertorié dans les titres de sections du fichier de configuration dont le nom contient le même code produit et la même version de produit que les paramètres d'entrée.

5. Le scanner collecte les propriétés de prérequis réelles pour les contrôles de prérequis à l'aide des collecteurs Prerequisite Scanner.
6. Le scanner vérifie les propriétés de prérequis dans le fichier de configuration associé au code produit et à la version du produit.

   Le scanner vérifie le système d'exploitation actuel en fonction du système d'exploitation pris en charge attendu dans la propriété de version de système d'exploitation prérequise ou dans les titres de sections du fichier de configuration dont le nom contient le même code produit et la même version de produit que les paramètres d'entrée.
7. Le scanner lit les propriétés de prérequis provenant du fichier de configuration et analyse les valeurs réelles et attendues des propriétés de prérequis pour les contrôles de prérequis. Il utilise les évaluateurs Prerequisite Scanner si nécessaire.
8. Le scanner affiche les résultats de l'analyse dans l'interface de ligne de commande, le texte des résultats et les fichiers XML ainsi que les fichiers journaux lisibles.
9. Le scanner nettoie et supprime les fichiers et répertoires temporaires.

Nouveautés dans cette édition

IBM Prerequisite Scanner version 1.2 propose de nouvelles propriétés et de nouvelles améliorations. Elle contient également des correctifs des incidents.

Nouvelles fonctionnalités de cette modification

Capacité à analyser et lire le nouveau fichier XML de résultats d'analyse.

Le kit d'outils Prerequisite Scanner Java Developer est un ensemble d'interfaces API qui permet aux développeurs de logiciels d'analyser et de lire à l'aide d'un programme le contenu du fichier XML de résultats pour répondre à leurs besoins ; par exemple analyser les résultats du balayage à utiliser dans un programme d'installation. Voir «Kit d'outils Java Developer de Prerequisite Scanner», à la page 35.

Nouveaux fichiers de configuration de cette modification

tableau 9 présente brièvement les nouveaux fichiers de configuration et codes produit fournis avec Prerequisite Scanner version 1.2.

Tableau 9. Nouveaux fichiers de configuration

Produit ou composant	Code produit	Fichier de configuration
Tivoli Composite Application Manager Agent for WebSphere MQ	KMQ	*ips_root*/Windows|UNIX_Linux/KMQ_07010000.cfg
Tivoli pourWebSphere Message Broker	KQI	*ips_root*/Windows|UNIX_Linux/KQI_07010000.cfg

Nouvelles propriétés de prérequis de cette modification

La propriété `os.SeaMonkeyVersion` a été ajoutée afin de vérifier la présence de la version de Mozilla SeaMonkey sur la machine. Voir «Propriétés de données du système d'exploitation», à la page 101.

La propriété `env.var.set.`*`env_var_name`* a été ajoutée afin de savoir si la variable d'environnement telle qu'indiquée par *env_var_name* est définie sur la machine. Voir «Propriétés de données de la variable d'environnement», à la page 115.

Améliorations de cette modification

Capacité à écrire les résultats d'analyse dans un fichier XML.

ips_output_dir`/result.xml` est le nouveau fichier de résultats d'analyse au format XML. Par défaut, l'outil affiche les résultats dans le fichier de résultats en texte brut uniquement. Voir «Formats de sortie», à la page 26.

`xmlResult` est un nouveau paramètre d'entrée facultatif du script Prerequisite Scanner de Prerequisite Scanner version 1.2 . Vous pouvez l'utiliser pour indiquer à l'outil d'afficher les résultats dans le fichier de résultats XML en plus du fichier de résultats de texte brut. Voir «prereq_checker», à la page 67.

Suppression de la section cumulée des résultats si ni les propriétés de prérequis de la mémoire, ni celles de l'espace disque n'existent dans les fichiers de configuration.
Prerequisite Scanner n'affiche plus les sections cumulées dans le fichier de résultats si ni les propriétés de prérequis de la mémoire, ni celles de l'espace disque n'existent dans les fichiers de configuration. Voir «Formats de sortie», à la page 26.

Fonctions obsolètes de cette modification

Aucun

Défauts corrigés de cette modification

Pour consulter la liste des incidents corrigés dans cette version, ouvrez le fichier `Readme.html` dans le répertoire *ips_root* lorsque vous extrayez le contenu des progiciels Prerequisite Scanner.

Changements apportés à la documentation dans cette modification

Le guide d'utilisateur Prerequisite Scanner n'est plus fourni avec les progiciels Prerequisite Scanner pour Prerequisite Scanner. Vous pouvez utiliser le centre de documentation IBM Prerequisite Scanner.

Chapitre 2. Installation de Prerequisite Scanner

Il n'y a pas de programme d'installation pour IBM Prerequisite Scanner. Lorsque vous extrayez le contenu du fichier compressé, les fichiers core se trouvent dans le répertoire principal avec les sous-répertoires suivants : `/api` pour le kit d'outils Prerequisite Scanner Java Developer prenant en charge l'interface API XML de la requête, `/lib` pour les collecteurs et scripts communs, `/Windows` pour les évaluateurs et fichiers de configuration sous Windows, `/UNIX_Linux` pour les évaluateurs et fichiers de configuration sur les plateformes UNIX et `/licenses` pour les fichiers de license.

Prérequis

IBM Prerequisite Scanner peut être exécuté sur les systèmes Windows, Windows XP ou ultérieurs, 32 bits ou 64 bits. Il peut également être exécuté sur des variantes des systèmes d'exploitation AIX, HP-UX, Linux et Solaris.

Vérifiez que vous avez installé les utilitaires suivants ou qu'ils sont disponibles dans les environnements cible :

Système cible	Prérequis
Windows	• Le client Telnet est activé de telle sorte que les contrôles de connectivité dans le collecteur de connectivité prédéfini peuvent s'effectuer correctement. • Assurez-vous que Microsoft .Net Framework 1.0 ou suivant est installé, afin que Prerequisite Scanner puisse fonctionner correctement.

Système cible	Prérequis

UNIX	• Bash est installé de telle sorte que les collecteurs Prerequisite Scanner UNIX peuvent fonctionner correctement.
	• Pour les utilisateurs non superutilisateurs, l'emplacement des commandes **mount**, **swapinfo** et **psrinfo** doit être défini dans la variable d'environnement `PATH` de telle sorte que les commandes sont disponibles pour Prerequisite Scanner. Les commandes se trouvent dans le répertoire `/usr/sbin` ; par exemple, définissez la variable d'environnement `PATH` comme suit :
	`export PATH=$PATH:/usr/sbin/`
	• Vérifiez que les autorisations d'accès affectées à la commande **lscfg** sont correctes, y compris les autorisations spécifiques définies par les indicateurs de droit d'accès, par exemple le bit `setuid`. Les autorisations d'accès correctes signifient que Prerequisite Scanner peut exécuter la commande et récupérer les informations concernant le système. La commande se trouve dans le répertoire `/usr/sbin` ; par exemple, pour définir le bit `setuid` de **lscfg**, exécutez la commande **chmod** comme suit :
	`chmod 4777 /usr/sbin/lscfg`

Prerequisite Scanner prend en charge tous les matériels et systèmes d'exploitation du produit spécifié ou la solution IBM pour laquelle vous exécutez Prerequisite Scanner.

Installation du fichier compressé

Vous pouvez extraire le contenu du fichier compressé de IBM Prerequisite Scanner. Vous devez disposer des droits en écriture pour le répertoire principal dans lequel vous extrayez le contenu du fichier compressé.

Procédure

1. Ouvrez votre navigateur Web et entrez l'adresse URL de IBM Fix Central. Vérifiez que vous avez bien ouvert `IBM.com` ou le portail de support IBM.
2. Dans la liste **Product Group (Groupe de produits)**, sélectionnez **Tivoli**.
3. Dans la liste **Product (Produit)**, sélectionnez IBM Prerequisite Scanner.
4. Dans la liste **Installed Version (Version installée)**, sélectionnez la version que vous souhaitez télécharger.
5. Dans la liste **Platform (Plateforme)**, sélectionnez la plateforme sur laquelle vous souhaitez installer Prerequisite Scanner.
6. Cliquez sur **Continuer**. La page Identify Fixes (Identifier les correctifs) s'ouvre.
7. Utilisez l'option par défaut **Browse for fixes (Rechercher les correctifs)** et cliquez sur **Continue (Poursuivre)**.

8. Sur la page Select fixes (Sélectionner les correctifs), sélectionnez le module, puis cliquez sur **Continue (Poursuivre)**.
9. Sur la page Download option (Option de téléchargement), sélectionnez l'option de téléchargement, puis cliquez sur **Download now (Télécharger maintenant)**.
10. Extrayez le contenu des fichiers compressés vers votre emplacement préféré comme indiqué par *ips_root*.

Que faire ensuite

Recherchez dans la documentation d'installation ou les notes techniques de votre produit les étapes supplémentaires qui doivent être réalisées avant d'exécuter Prerequisite Scanner. Par exemple, vous pouvez définir la variable d'environnement indiquant à Prerequisite Scanner les composants ou fonctionnalités installées sur l'ordinateur cible et, par conséquent, les prérequis à contrôler.

Désinstallation de Prerequisite Scanner

Supprimez IBM Prerequisite Scanner si vous souhaitez installer une version plus récente, le déplacer vers un autre environnement ou il s'agit d'une version dont vous n'avez plus besoin.

Procédure

1. Ouvrez le répertoire *ips_root*.
2. Supprimez le répertoire et son contenu.

Chapitre 3. Extension de Prerequisite Scanner

IBM Prerequisite Scanner fournit un ensemble de base de collecteurs, d'évaluateurs et de configurations que vous pouvez utiliser pour exécuter l'outil et rechercher des prérequis. Si l'ensemble de base des fichiers, les propriétés et valeurs de prérequis et les contrôles de prérequis ne répondent pas à vos exigences, vous pouvez étendre Prerequisite Scanner.

Avant d'exécuter Prerequisite Scanner

Avant d'exécuter IBM Prerequisite Scanner, déterminez si les propriétés de prérequis prédéfinies, leurs valeurs attendues et les fichiers de configuration répondent à vos exigences en matière d'analyse des prérequis. Si l'un de ces éléments ne répond pas à vos besoins, vous pouvez exécuter un ensemble de tâches de prérequis pour configurer ou étendre Prerequisite Scanner. L'ensemble de contrôles et de tâches de prérequis dépend de la plateforme et du nombre de contrôles de prérequis.

Contrôles obligatoires et tâches d'extension pour les systèmes Windows

Vous devez exécuter un ensemble de contrôles et tâches avant d'exécuter IBM Prerequisite Scanner. Ces contrôles déterminent si vous pouvez modifier et utiliser les fichiers de configuration existants ou si vous devez étendre Prerequisite Scanner.

tableau 10 établit une liste de contrôles et de tâches à exécuter.

Tableau 10. Contrôles et tâches à exécuter avant d'utiliser un fichier de configuration pour les systèmes Windows

	Contrôle	**Tâche**
☐	Vérifiez que le produit, ses systèmes d'exploitation pris en charge et les versions du système d'exploitation sont répertoriés dans le fichier `codename.cfg`.	• Si oui, effectuez le contrôle suivant. • Si non, ajoutez au fichier un code produit, un système d'exploitation individuel et la version du système d'exploitation facultatif. Pour plus d'informations, voir «Ajout de codes produit», à la page 47.
☐	Vérifiez qu'il existe un fichier de configuration pour le code produit associé à la version du produit.	• Si oui, effectuez le contrôle suivant. • Si non, créez un fichier de configuration contenant les propriétés de prérequis pour ce système d'exploitation et la version du système d'exploitation. Pour plus d'informations, voir «Créer des fichiers de configuration personnalisés», à la page 48.
☐	Ouvrez le fichier de configuration et vérifiez qu'il contient les bonnes propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, ajoutez des propriétés de prérequis. Pour plus d'informations, voir «Ajouter les propriétés de prérequis», à la page 50.

Tableau 10. Contrôles et tâches à exécuter avant d'utiliser un fichier de configuration pour les systèmes Windows (suite)

	Contrôle	Tâche
☐	Vérifiez que les propriétés de prérequis ont les valeurs attendues.	• Si oui, exécutez Prerequisite Scanner. Pour plus d'informations, voir Chapitre 4, «Exécution de Prerequisite Scanner», à la page 67. • Si non, modifiez les propriétés de prérequis. Pour plus d'informations, voir «Éditer des propriétés de prérequis», à la page 52.
☐	Pour toute nouvelle propriété de prérequis, vérifiez que les collecteurs prédéfinis peuvent collecter les valeurs réelles pour les propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, créez des collecteurs personnalisés. Pour plus d'informations, voir «Création de collecteurs personnalisés pour sysdtèmes Windows», à la page 52.
☐	Pour toute propriété de prérequis nouvelle ou modifiée, vérifiez que les évaluateurs prédéfinis peuvent comparer les valeurs attendue et réelle pour les propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, créez des évaluateurs personnalisés. Pour plus d'informations, voir «Créer des évaluateurs personnalisés pour les systèmes Windows», à la page 60.
☐	Vérifiez que tous les fichiers ont été enregistrés dans les bons répertoires : • fichiers de configuration, tous les collecteurs spécifiques au produit personnalisés et les fichiers de traitement par lots associés ainsi que tous les fichiers d'évaluateur personnalisés du répertoire *ips_root*`/Windows` • collecteurs communs personnalisés du répertoire *ips_root*`/lib`	Exécutez Prerequisite Scanner. Pour plus d'informations, voir Chapitre 4, «Exécution de Prerequisite Scanner», à la page 67.

Contrôles et tâches d'extension requis pour les systèmes UNIX

Vous devez exécuter un ensemble de contrôles et tâches de prérequis avant d'exécuter IBM Prerequisite Scanner. Ces contrôles déterminent si vous pouvez modifier et utiliser les fichiers de configuration existants ou si vous devez étendre Prerequisite Scanner.

tableau 11 établit une liste de contrôles et de tâches obligatoires à exécuter.

Tableau 11. Contrôles et tâches à exécuter avant d'utiliser un fichier de configuration pour les systèmes UNIX

	Contrôle	Tâche
☐	Vérifiez que le produit est répertorié dans le fichier `codename.cfg`.	• Si oui, effectuez le contrôle suivant. • Si non, ajoutez un code produit au fichier `codename.cfg`. Pour plus d'informations, voir «Ajout de codes produit», à la page 47.
☐	Vérifiez qu'il existe un fichier de configuration pour le code produit associé au produit.	• Si oui, effectuez le contrôle suivant. • Si non, créez un fichier de configuration contenant les propriétés de prérequis pour toutes les plateformes prises en charge du produit. Pour plus d'informations, voir «Créer des fichiers de configuration personnalisés», à la page 48.

Tableau 11. Contrôles et tâches à exécuter avant d'utiliser un fichier de configuration pour les systèmes UNIX (suite)

	Contrôle	Tâche
☐	Ouvrez le fichier de configuration et vérifiez qu'il contient les bonnes propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, ajoutez des propriétés de prérequis. Pour plus d'informations, voir «Ajouter les propriétés de prérequis», à la page 50.
☐	Vérifiez que les propriétés de prérequis ont les valeurs attendues.	• Si oui, exécutez Prerequisite Scanner. Pour plus d'informations, voir Chapitre 4, «Exécution de Prerequisite Scanner», à la page 67. • Si non, modifiez les propriétés de prérequis. Pour plus d'informations, voir «Éditer des propriétés de prérequis», à la page 52.
☐	Pour toute nouvelle propriété de prérequis, vérifiez que les collecteurs prédéfinis peuvent collecter les valeurs réelles pour les propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, créez des collecteurs personnalisés. Pour plus d'informations, voir «Création de collecteurs personnalisés pour systèmes UNIX», à la page 57.
☐	Pour toute propriété de prérequis nouvelle ou modifiée, vérifiez que les évaluateurs peuvent comparer les valeurs attendue et réelle pour les propriétés de prérequis.	• Si oui, effectuez le contrôle suivant. • Si non, créez des évaluateurs personnalisés. Pour plus d'informations, voir «Création d'évaluateurs personnalisés pour les systèmes UNIX», à la page 64.
☐	Pour toute propriété de prérequis nouvelle ou modifiée, vérifiez que le code permettant d'appeler et d'exécuter les collecteurs figure dans le script *ips_root*/`UNIX_Linux/packageTest.sh`.	• Si oui, effectuez le contrôle suivant. • Si non, modifiez le script de test de module maître. Pour plus d'informations, voir «Éditer un script de test de package pour les systèmes UNIX», à la page 58.
☐	Vérifiez que tous les fichiers ont été enregistrés dans les bons répertoires : • fichiers de configuration, tous les fichiers de collecteur personnalisés et tous les fichiers d'évaluateur personnalisés du répertoire *ips_root*/`UNIX_Linux`	Exécutez Prerequisite Scanner. Pour plus d'informations, voir Chapitre 4, «Exécution de Prerequisite Scanner», à la page 67.

Ajout de codes produit

IBM Prerequisite Scanner fournit un ensemble de codes prédéfinis de version produit dans le fichier `codename.cfg`. Vous pouvez ajouter des codes produit si le fichier n'en dispose pas pour la version du produit, pour ses plateformes prises en charge et pour les versions des systèmes d'exploitation

Procédure

1. Ouvrez le fichier *ips_root*/`codename.cfg`.
2. Vérifiez si le fichier contient déjà des paires de valeurs de nom pour les versions de produits.
3. Si le code produit n'existe pas, ajoutez-en un et assurez-vous que vous utilisez le format correct comme suit :

   ```
   product_code=code_value
   ```

   Restriction : IBM Tivoli Monitoring et Tivoli Composite Application Manager disposent de codes produit prédéfinis que Prerequisite Scanner considère comme réservés. Ces codes ne doivent pas être utilisés en tant que codes

produit Prerequisite Scanner à moins qu'ils fassent référence à leurs agents associés IBM Tivoli Monitoring et Tivoli Composite Application Manager. Pour en savoir plus sur ces codes produit, voir ITM 6.X Product Codes Technote.

Restriction : Sur UNIX seulement : lorsque vous entrez la valeur du code produit dans le fichier, évitez d'utiliser `for`. C'est un mot réservé qui peut avoir un impact sur le fonctionnement de Prerequisite Scanner.

Par exemple, pour ajouter un code produit pour IBM Tivoli Monitoring for Energy Management sur toutes les plateformes Windows, ajoutez la ligne suivante au fichier :

```
MEA=IBM Tivoli Monitoring for Energy Management
```

Créer des fichiers de configuration personnalisés

Vous pouvez créer des fichiers de configuration personnalisés à partir de l'échantillon fichier de configuration si les configurations prédéfinies ne répondent pas à vos exigences de propriétés de prérequis. Avant de créer un fichier de configuration personnalisé, assurez-vous que vous connaissez les propriétés de prérequis que vous souhaitez ajouter et leurs valeurs escomptées.

Pourquoi et quand exécuter cette tâche

Important : Vous devez inscrire les conventions d'attribution et les règles de formatage qui régissent la création et l'édition d'un fichier de configuration personnalisé. Sinon, Prerequisite Scanner ne peut pas exécuter un scan avec succès en utilisant ce fichier.

Procédure

1. Ajoutez, si nécessaire, des codes produits pour le produit au fichier `codename.cfg`.
2. Créez le fichier de configuration en utilisant l'éditeur de texte dans le répertoire *ips_root/OS*. Assurez-vous que la convention d'attribution de nom pour le nom du fichier :

   product_code_version`.cfg`

   où :

   - *product_code*

     Il s'agit de la variable représentant un code produit sur les systèmes Windows ou UNIX. Les codes produit identifient le produit, une plateforme individuelle, telle que Windows, AIX, HP-UX, Linux et Solaris, et éventuellement la version du système d'exploitation prise en charge par ce produit. Ils sont stockés dans le fichier `codename.cfg`. Tout produit prenant en charge plusieurs plateformes comprend plusieurs codes produit, chacun identifiant un produit, une plateforme et la version du système d'exploitation, selon les besoins.
   - *version* est le code à 8 chiffres représentant la version, l'édition, la modification et le niveau avec deux chiffres pour chaque partie du code. Par exemple : 7.3.21 est 07032100.
3. Revoyez les propriétés de prérequis de base indiquées dansAnnexe C, «Référence de propriétés de prérequis», à la page 91 et déterminez la propriété de prérequis que vous souhaitez vérifier.
4. Facultatif : Ajoutez une section et assurez-vous que vous utilisez la convention d'attribution de nom pour le titre de la section :
   - **Catégorie de type de données unique et prédéfinie**

```
[category_name :category_value]
```

Par exemple, pour créer une section pour des propriétés de prérequis communes à toutes les plates-formes Windows, ajoutez le titre de la section suivante :

```
[OSType:Windows]
```

Par exemple, pour créer une section pour des propriétés de prérequis communes à toutes les variantes OS Linux de RedHat, ajoutez le titre de la section suivante:

```
[OSType:RedHat]
```

- **Catégories de type de données combinées, prédéfinies**

```
[category_name :category_value]
[category_name :category_value]
```

Par exemple, pour créer une section pour les propriétés de prérequis des variantes du serveur Windows 2003 sauf la variante R2 du serveur Windows 2003, ajoutez le titre de la section de combinaison suivante :

```
[OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]
```

Par exemple, créez une section de propriétés de prérequis pour SUSE Linux Enterprise Server 11 OS et si la variable environnement `@TPAE_DB_SERVER` est définie à `true`. Ajoutez le titre de la section de combinaison suivante :

```
[OSType=SUSELinuxEnterpriseServer][@TPAE_DB_SERVER:true]
```

où :

category_name est le code multi-caractères qui représente la catégorie de type de données comme indiqué dans tableau 6, à la page 17

category_value est le code multi-caractères qui représente une valeur autorisée pour la catégorie comme indiqué dans tableau 6, à la page 17

5. Facultatif : Pour chaque section, revoyez les propriétés de prérequis de base indiquées dansAnnexe C, «Référence de propriétés de prérequis», à la page 91 et déterminez la propriété de prérequis que vous souhaitez vérifier.
6. Pour chaque propriété de prérequis que vous souhaitez ajouter, saisissez un nom de paire de valeurs avec des qualificateurs optionnels comme demandé. Assurez-vous que vous utilisez le format suivant avec une seule propriété de prérequis sur chaque ligne :

```
[prefix_identifier.]property_name[.suffix_identifier]=
[qualifier_name :qualifier_value]property_value
```

où :

- *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis comme indiqué danstableau 3, à la page 4. Cet identificateur préfixe est requis par quelques unes des catégories prédéfinies.
- *property_name* est le nom de la propriété de prérequis.
- *suffix_identifier* est un identificateur optionnel pour un sous-type de propriétés de prérequis comme indiqué dans tableau 4, à la page 7.
- *qualifier_name* est un attribut optionnel de la propriété de prérequis. IBM Prerequisite Scanner l'utilise pour qualifier la propriété de prérequis ou le type de vérification pour exécuter la propriété de prérequis, comme indiqué dans «Catégories prédéfinies des propriétés de prérequis», à la page 9.

  Remarque : Vous pouvez avoir plusieurs qualificateurs, chacun séparé par une virgule. L'ensemble des qualificateurs doit figurer entre crochets`[]`.

- *qualifier_value* est la valeur pour l'attribut optionnel. Chaque qualificateur et sa valeur doivent être séparés par deux points **:** colon.
- *property_value* est la valeur pour la propriété de prérequis et peut être une chaîne ou un nombre entier.

Par exemple, la catégorie utilisateur prédéfinie despropriétés de prérequis dispose de l'identificateur préfixe `user`. La propriété de prérequiss pour vérifier si l'utilisateur connecté fait partie du groupe utilisateur administrateur est `:user.isAdmin=True`

7. Si une propriété de prérequis n'existe pas dans les catégories prédéfinies , ajoutez le nom de la propriété de prérequis personnalisé, sa valeur et ses qualificateurs optionnels. Vous devez alors créer les fichiers suivants pour vérifier et comparer les propriétés de prérequis comme demandé : un collecteur personnalisé pour collecter la valeur réelle pour la propriété prérequise et l'évaluateur personnalisé si les fonctions standards de comparaison ne peuvent pas comparer les valeurs réelles et escomptées.

Ajouter les propriétés de prérequis

Vous pouvez ajouter des propriétés de prérequis de base à partir des catégories prédéfinies pour les propriétés de prérequis aux fichiers de configuration. Vous pouvez également ajouter des propriétés de prérequis personnalisées.

Pourquoi et quand exécuter cette tâche

Important : Vous devez adhérer aux règles de formatage régissant l'ajout et l'édition de propriétés de prérequis à un fichier de configuration. Sinon, Prerequisite Scanner ne peut pas exécuter un scan avec succès pour cette propriété de prérequis.

Procédure

1. Ouvrez le fichier de configuration.
2. Revoyez les propriétés de prérequis de base indiquées dansAnnexe C, «Référence de propriétés de prérequis», à la page 91 et déterminez la propriété de prérequis que vous souhaitez vérifier.
3. Pour chaque propriété de prérequis que vous souhaitez ajouter, saisissez un nom de paire de valeurs avec des qualificateurs optionnels comme demandé.

   Par exemple, pour ajouter des propriétés de prérequis à partir de la catégorie commune prédéfinie, saisissez le nom de la propriété et l'unique valeur escomptée. Ajoutez les propriétés de prérequis suivantes au fichier :

   ```
   Disk=1GB
   OS Version=regex{Windows 200[3-8]}
   ```

   Par exemple, la catégorie réseau prédéfinie des propriétés de prérequis dispose de l'identificateur préfixe `network` et le nom de la propriété de prérequis pour vérifier les ports est `availablePorts`. Vous pouvez, par ailleurs, catégoriser les ports disponibles par sous-types d'applications `DB2` pour le serveur de base de données DB2, pour Websphere`WAS`Web`Sphere`FTP Application Server et pour le protocole FTP. Ajoutez les propriétés de prérequis suivantes au fichier :

   ```
   network.availablePorts.DB2=5000-5005
   network.availablePorts.WAS=9080
   network.availablePorts.FTP=21
   ```

   Par exemple, la catégorie système d'exploitation prédéfinie des propriétés de prérequis a l'identificateur préfixe `os` et le nom de la propriété de prérequis pour vérifier l'espace disque disponible pour les systèmes du fichier est`space`.

Vous pouvez en outre catégoriser les sous-types de système de vérification par fichier, `usr` `ethome`. Vous pouvez spécifier les valeurs pour le `dir` et les qualificateurs `unit`.

Ajoutez les propriétés de prérequis suivantes au fichier :

```
os.space.usr=[dir:root=/usr/ibm/common/acsi,unit:GB]2
os.space.home=[dir:non_root=USERHOME/.acsi_HOST,unit:MB]200
```

Important : Vous ne pouvez utiliser les qualificateurs qu'avec des propriétés de prérequis prédéfinies spécifiques, comme indiqué dans tableau 5, à la page 10.

4. Si une propriété de prérequis n'existe pas dans les catégories prédéfinies de la propriété de prérequis, ajoutez le nom de la paire de valeurs avec le qualificateur optionnel pour la propriété de prérequis personnalisée et la valeur. Assurez-vous que vous utilisez le format suivant avec une seule propriété de prérequis sur chaque ligne.

   [*prefix_identifier*.]*property_name*[.*suffix_identifier*]=
   [[*qualifier_name* :*qualifier_value*]]*property_value*

   où :

   - *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis comme indiqué dans tableau 3, à la page 4. Cet identificateur préfixe est requis par quelques unes des catégories prédéfinies.
   - *property_name* est le nom de la propriété de prérequis.
   - *suffix_identifier* est un identificateur optionnel pour un sous-type de propriétés de prérequis comme indiqué dans tableau 4, à la page 7.
   - *qualifier_name* est un attribut optionnel de la propriété de prérequis. IBM Prerequisite Scanner l'utilise pour qualifier la propriété de prérequis ou le type de vérification pour exécuter la propriété de prérequis, comme indiqué dans «Catégories prédéfinies des propriétés de prérequis», à la page 9.

     Remarque : Vous pouvez avoir plusieurs qualificateurs, chacun séparé par une virgule. L'ensemble des qualificateurs doit figurer entre crochets`[]`.
   - *qualifier_value* est la valeur pour l'attribut optionnel. Chaque qualificateur et sa valeur doivent être séparés par deux points `:`.
   - *property_value* est la valeur pour la propriété de prérequis et peut être une chaîne ou un nombre entier.

   Par exemple, `env.tcrhome` est une propriété de prérequis personnalisée qui vérifie l'environnement de la variable répertoire de base pour Tivoli Common Reporting, et la valeur escomptée doit être `True`:

   ```
   env.tcrhome=True
   ```

   `env.path.jar` est une propriété de prérequis qui vérifie si l'environnement d'exécution Java est défini dans l'environnement de la variable `PATH`, et la valeur escomptée doit être `False`:

   ```
   env.path.jar=False
   ```

   Remarque : Vous devez alors créer les fichiers suivants pour vérifier et comparer les propriétés de prérequis comme demandé : un collecteur personnalisé pour collecter la valeur réelle pour la propriété prérequise et l'évaluateur personnalisé uniquement si les fonctions standards de comparaison ne peuvent pas comparer les valeurs réelles et escomptées.

Éditer des propriétés de prérequis

Vous pouvez éditer des propriétés de prérequis, modifier la valeur escomptée pour ces propriétés de prérequis ou encore modifier les valeurs associées des qualificateurs.

Avant de commencer

Vérifier si la nouvelle valeur est une valeur valide qui est prise en charge par une propriété de prérequis. Par exemple, la propriété de prérequis `Disk` attend un format numérique avec soit l'unité Mo ou l'unité Go. Si vous souhaitez vérifier l'espace disque disponible en téraoctets (To), vous devez étendre la comparaison API pour prendre en charge les comparaisons To. Vous devez également modifier la propriété de prérequis `Disk` dans les fichiers de configuration concernés.

Vérifiez les qualificateurs prédéfinis et les valeurs valides pour la propriété de prérequis, comme indiqué dans «Catégories prédéfinies des propriétés de prérequis», à la page 9.

Procédure

1. Ouvrez le fichier de configuration.
2. Pour chaque propriété de prérequis que vous souhaitez éditer, saisissez la nouvelle valeur escomptée ou modifiez la valeur pour le qualificateur. Par exemple, un nouvel administrateur système est l'utilisateur principal, donc la valeur de la propriété requise `user.userID` doit changer. Modifiez la valeur dans le nouveau nom :

   ```
   user.userID=smithj
   ```

   Par exemple, le qualificateur `type` de la propriété de prérequis `os.ulimit` dispose actuellement d'une valeur de `filedescriptorlimit` pour vérifier les limites pour les descripteurs de fichiers. Vous souhaiteriez vérifier une autre limite, par exemple : l'espace mémoire. Modifiez les qualificateurs suivants pour la propriété de prérequis de :

   ```
   os.ulimit=[type:filedescriptorlimit]8192+,unlimited
   ```

   à :

   ```
   ios.ulimit=[type:stacksizelimit]512+, illimité
   ```

   Important : Vous ne pouvez utiliser les qualificateurs qu'avec des propriétés de prérequisprédéfinies spécifiques, comme indiqué danstableau 5, à la page 10.

Création de collecteurs personnalisés pour sysdtèmes Windows

Vous pouvez créer des connecteurs personnalisés si l'ensemble des collecteurs de base n'effectuent pas de collecte de valeurs pour les propriétés de prérequis nécessaires au produit devant être installé. Vous pouvez créer des collecteurs VBScript communs et personnalisés pour collecter des données pour des propriétés de prérequis qui s'appliquent à tout produit et à toute version de produit. Autrement, vous pouvez créer des collecteurs spécifiques au produit pour collecter des données qui s'appliquent à un produit spécifique et à une version du produit. Alors que chaque type de collecteur VBScript personnalisé collecte des données en utilisant les mêmes méthodes, les règles de création, de stockage, et d'exécution sont légèrement différentes.

Création de collecteurs VBScript personnalisés communs à tous les fichiers de configuration

Lorsque vous créez des collecteurs VBScript communs et personnalisés, le nom du fichier doit contenir le nom de la propriété de prérequis et doit être stocké dans le `/sous-repertoire` lib. Le collecteur contient le code permettant d'obtenir la valeur réelle pour une propriété de prérequis. Il peut également utiliser les fonctions communes et les sous-routines pour obtenir la valeur si nécessaire.

Avant de commencer

Assurez-vous d'avoir revu l'ensemble des fonctions prédéfinies et les sous-routines dans les annexes suivantes avant de créer les collecteurs. Déterminez si vous pouvez utiliser chacun d'entre eux afin d'obtenir les valeurs réelles:

- Annexe E, «Fonctions communes pour les systèmes Windows», à la page 123
- Annexe G, «Utilitaire de fichiers sous-routines pour les systèmes Windows», à la page 139
- Annexe F, «Sous-routines des utilitaires de journalisation pour les systèmes Windows», à la page 137
- Annexe H, «Autres fonctions communes et sous-routines pour les systèmes Windows», à la page 141

Déterminez si le collecteur doit vérifier que la propriété de prérequis existe et si elle n'existe pas, quelle autre information doit être recueillie. Chaque vérification doit renvoyer une valeur, si elle existe ou pas. Par exemple:

- vérifier si une variable d'environnement existe, comme le répertoire de base du produit, par exemple `TCR_HOME` pour Tivoli Common Reporting.
- vérifiez si la variable d'environnement contient un fichier JAR, binaire, ou chemin d'accès, comme le chemin d'accès au JRE dans la `PATH` variable d'environnement.
- vérifiez la valeur réelle de la variable d'environnement, comme le répertoire de base du produit, par exemple `TCR_HOME` pourTivoli Common Reporting.
- vérifiez si le produit est installé.
- vérifiez quelle version du produit est installée.

Procédure

1. Créer un fichier VBScript. Enregistrer le fichier dans le *ips_root*`/répertoire` lib, avec la variante de la convention d'attribution de nom du fichier suivante:

   `[`*`prefix_identifier`*`.]property_name.vbs`

   où :

   - *prefix_identifier* est l'identificateur à préfixe pour une catégorie prédéfinie de propriétés de prérequis comme décrit dans tableau 3, à la page 4.
   - *property_name* est le nom de la propriété de prérequis et utilisé dans le nom du collelcteur.

   Par exemple, `mssqlVersion.vbs` contient le code permettant d'obtenir la valeur réelle pour le serveur MS SQL de la propriété de prérequis dans la machine Windows.
2. Utiliser un éditeur VBScript, permet au code d'obtenir la valeur pour la propriété de prérequis. Utilisez une COM VBScript et les fonctions pour

accéder aux éléments de l'environnement Windows et les exécuter dansl'environnement hôte script Windows. Assurez-vous que la vérification renvoie la sortie standard comme suit :

```
WScript.Echo "property_name=" &#38; var_for_value
```

- *property_name* qui représente la propriété de prérequis comme écrit dans le fichier de configuration, par exemple, `env.tcrhome`.
- *var_for_value*, qui est, la variable VBScript pour la valeur réelle obtenue par le collecteur pour la propriété de prérequis.

Pour vérifier si `TCR_HOME` l'environnement existe et renvoie la valeure réelle, pour laquelle le nom de la propriété de prérequis est `env.tcrhome`:

```
set wshShell = WScript.CreateObject("WScript.Shell")
tcr_home=WshShell.ExpandEnvironmentStrings("%TCR_HOME%")
WScript.Echo "env.tcrhome=" &#38; tcr_home
```

Pour vérifier si le JRE est défini dans le chemin d'accès à la variable, pour laquelle le nom de la propriété de prérequis est `env.path.jre`:

```
Définir wshShell = WScript.Créer un objet("WScript.Shell")
Chemin d'accès = WshShell.ExpandEnvironmentStrings("%PATH%")
Définir objRegEx = nouveau RegExp
objRegEx.Pattern = "(^|([:;\\/]))(C:\Program Files\IBM\Java60\jre\bin)($|[:;])"
objRegEx.IgnoreCase = True
objRegEx.Global = True
Définir des correspondances = objRegEx.Execute(path)
WScript.Echo "env.path.jre=" &#38; (matches.Count > 0)
```

Pour vérifier la version de Tivoli Directory Integrator installée, où le nom de la propriété de prérequis se trouve `installedSoftware.TDI.version`:

```
strComputer = "."
strKeyPath = "SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall"
regDisName = "DisplayName"
regDisVer = "DisplayVersion"

Set oReg = GetObject("winmgmts:{impersonationLevel=Impersonate}!\\" &#38;
          strComputer &#38; "\root\default:StdRegProv")

Set sftReg = new RegExp
sftReg.pattern = "Tivoli Directory Integrator"
sftReg.Global=False
oReg.EnumKey HKEY_LOCAL_MACHINE, strKeyPath, arrSubKeys
For Each subkey In arrSubKeys
    searchkey = strKeyPath &amp; "\" &#38; subkey
    oReg.GetStringValue HKEY_LOCAL_MACHINE, searchkey, regDisName, strName
    oReg.GetStringValue HKEY_LOCAL_MACHINE, searchkey, regDisVer, strVersion
    If Not IsNull(strName) Then
       Set matches = sftReg.Execute(strName)
       If matches.Count > 0 Then
           Wscript.Echo "installedSoftware.TDI.version=" &#38; strVersion
       End If
    End If
Next
```

3. Exécutez le collecteur VBScript afin de s'assurer qu'il n'existe pas d'erreurs d'exécution et de débogage si nécessaire.
4. Créez un évaluateur personnalisé uniquement si les fonctions de comparaison standard ne peuvent pas comparer les valeurs réelles et prévues.

Création de collecteurs VBScript personnalisés et spécifiques à un produit et à une version du produit

Lorsque vous créez des collecteurs VBScript personnalisés et spécifiques à un produit, le nom du fichier doit être le même code produit que celui du fichier de configuration et stocké dans le `/sous-répertoire` Windows. Le collecteur peut contenir le code permettant de recueillir les valeurs réelles pour une ou plusieurs propriétés de prérequis. Il peut également utiliser les fonctions communes et les sous-routines pour recueillir ces valeurs si nécessaire.

Avant de commencer

Assurez-vous d'avoir revu l'ensemble des fonctions et les sous-routines dans les annexes suivantes avant de créer les collecteurs. Déterminez si vous pouvez utiliser chacun d'entre eux afin d'obtenir les valeurs réelles:

- Annexe E, «Fonctions communes pour les systèmes Windows», à la page 123
- Annexe G, «Utilitaire de fichiers sous-routines pour les systèmes Windows», à la page 139
- Annexe F, «Sous-routines des utilitaires de journalisation pour les systèmes Windows», à la page 137
- Annexe H, «Autres fonctions communes et sous-routines pour les systèmes Windows», à la page 141

Déterminez si le collecteur doit vérifier que la propriété de prérequis existe et si elle n'existe pas, quelle autre information doit être recueillie. Chaque vérification doit renvoyer une valeur, si elle existe. Par exemple:

- vérifiez si le répertoire existe.
- vérifiez l'espace disque disponible pour le répertoire.
- vérifiez si un produit est installé.
- vérifiez quelle version du produit est installée.

Procédure

1. Création d'un fichier VBScript. Enregistrez le fichier dans le *ips_root*`/répertoire` Windows, avec la variante de la convention d'attribution de nom du fichier suivante :

   product_code[*_version*]`.vbs`

   où :

   - *product_code*

     Il s'agit de la variable représentant un code produit sur les systèmes Windows ou UNIX. Les codes produit identifient le produit, une plateforme individuelle, telle que Windows, AIX, HP-UX, Linux et Solaris, et éventuellement la version du système d'exploitation prise en charge par ce produit. Ils sont stockés dans le fichier `codename.cfg`. Tout produit prenant en charge plusieurs plateformes comprend plusieurs codes produit, chacun identifiant un produit, une plateforme et la version du système d'exploitation, selon les besoins.
   - *version* est le code à 8-chiffres pour représenter la version, la publication, les modifications, et le niveau, avec deux chiffres pour chaque partie du code; par exemple, 7.3.21 est 07032100.
2. Utilisation d'un éditeur VBScript, ouvrez le fichier et incluez le chemin d'accès à `common_function.vbs` si vous devez utiliser des fonctions communes, comme suit :

```
Include("..\lib\common_function.vbs")
```

3. Si vous devez utiliser les valeurs de `PATH` et `-p` l'indicateur passé à partir de Prerequisite Scanner, puis utilisez`Wscript.Arguments()` où `Wscript.Arguments(0)` la valeur se trouve pour `PATH`. `Wscript.Arguments(1)` est `-p` l'indicateur et ses valeurs.
4. Ajoutez le code afin d'obtenir la valeur pour la propriété de prérequis en utilisant COM VBScript et les fonctions pour accéder aux éléments de l'environnement Windows. Exécutez les dans l'environnement Windows script host. Assurez-vous que la vérification renvoie la sortie standard comme suit :

   `WScript.Echo "`*`property_name`*`=" &` *`var_for_value`*

   - *property_name* qui représente la propriété de prérequis comme écrit dans le fichier de cinfiguration, par exemple, `env.tcrhome`.
   - *var_for_value*, c'est-à-dire, la variable VBScript pour la valeur réelle obtenue par le collecteur pour la propriété de prérequis.

   Pour vérifier l'espace de disque disponible pour le répertoire d'installation du produit. Par exemple, vérifiez pour Tivoli Monitoring for Energy Management Reporting and Optimization en utilisant «getValue()», à la page 142 la sous-routine, où la propriété de prérequis se trouve `InstallDir`:

```
Set wshShell = WScript.CreateObject("WScript.Shell")
'Check the disk space for the installation path that is passed as
 the value for the PATH argument
installPath = Wscript.Arguments(0)
sInstallPath= "InstallDir="
Wscript.Echo "installation path    : " & installPath
set fso = CreateObject("Scripting.FileSystemObject")

getValue fso, sInstallPath, installPath

'Common sub routine
Sub getValue(fso, sKey, drvPath)
    Wscript.Echo "getValue(" & skey & "," & drvPath & ")"
   If fso.driveExists(fso.getDriveName(drvPath))  then
       Set disk = fso.GetDrive(fso.getDriveName(drvPath))
         'Value returned is in bytes.  Convert to MB
         cSize = CLng((disk.FreeSpace/1024)/1024) & "MB"
         WScript.Echo sKey & cSize
    Else
     Wscript.Echo " Disk for " & sKey & " -> " & drvPath & " does NOT exist"
   End If
End Sub
```

5. Créez un fichier de commande pour appeler le collecteur VBScript. Le fichier de commande doit avoir le même nom que le fichier de configuration et une `.bat` extension, *product_code*[*_version*]`.bat`, comme suit :

```
@echo off

set CMD_LINE_ARGS=
:setArgs
if ""%1""=="""" goto doneSetArgs
set CMD_LINE_ARGS=%CMD_LINE_ARGS% %1
shift
goto setArgs
:doneSetArgs

cscript.exe //nologo collector_file_name.vbs %CMD_LINE_ARGS%
```

6. Exécutez le collecteur VBScript afin de s'assurer qu'il n'existe pas d'erreurs d'exécution et de débogage si nécessaire.
7. Créez un évaluateur personnalisé uniquement si les fonctions de comparaison standard ne peuvent pas comparer les valeurs réelles et prévues.

Création de collecteurs personnalisés pour systèmes UNIX

Vous pouvez créer des collecteurs personnalisés si l'ensemble de collecteurs de base ne réalise pas de collecte pour les propriétés de prérequis nécessaires au produit à installer. Lorsque vous créez des collecteurs personnalisés, le nom de fichier doit être le même que celui de la propriété de prérequis quoique sans le sous-type en dans nom. Le collecteur est enregistré dans le `/sous-répertoire` UNIX_Linux. Le collecteur peut contenir le code permettant d'obtenir les valeurs réelles pour une ou plusieurs propriétés de prérequis. Il peut également utiliser les fonctions communes afin d'obtenir ces valeurs si nécessaire.

Avant de commencer

Assurez-vous d'avoir revu l'ensemble des fonctions dans les annexes suivantes avant de créer des collecteurs. Déterminez si vous pouvez utiliser chacun d'entre eux afin d'obtenir les valeurs réelles:

- Annexe I, «Fonctions communes pour les systèmes UNIX», à la page 145
- Annexe J, «Autres fonctions pour les systèmes UNIX», à la page 153
- Annexe K, «Fonctions de l'utilitaire de journalisation pour les systèmes UNIX», à la page 161

Déterminez si le collecteur doit vérifier que la propriété de prérequis existe et si elle n'existe pas, quelle autre information doit être reccueillie. Chaque vérification doit renvoyer une valeur, si elle existe. Par exemple:

- vérifiez si un produit est installé, par exemple, un module installé avec RPM.
- vérifiez quelle version du produit est installée.
- vérifiez l'espace de disque disponible pour le système de fichier monté

Si vous souhaitez utiliser des sous-types, *suffix_identifier*, et catégoriser d'avantage une propriété de prérequis par une application, un utilitaire, ou un sous-type de service, vous pouvez créer un collecteur commun. Passez le différentiateur pour le *suffix_identifier* sous-type, c'est-à-dire, *differentiator_suffix_identifier* à son collecteur. Par exemple, `os.package` est le collecteur commun pour vérifier l'existence des modules. Pour vérifier l'existence de `openssh`, passez le nom du module lors de l'appel du`os.package` collecteur dans le `le fichier script` du test de package, comme suit :

```
./os.package openssh
```

où `openssh` est le nom du module, c'est-à-dire, le *suffix_identifier* sous-type et le *differentiator_suffix_identifier* différentiateur.

Procédure

1. Créez un fichier script de shell. Enregistrez le fichier dans le *ips_root*/`répertoire` UNIX_Linux, avec la variante de la convention d'attribution de nom du fichier suivante mais sans extension de fichier:

   `[`*`prefix_identifier`*`.]`*`property_name`*

   Où :

   - *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis comme décrit dans tableau 3, à la page 4. Cet identificateur à préfixe est exigé par certaines des catégories prédéfinies, par exemple, `env`.
   - *property_name* est le nom de la propriété de prérequis, par exemple, `path.jre`

2. Utiliser un éditeur, ouvrir le fichier et inclure le chemin d'accès à `common_function.sh` si vous devez utiliser des fonctions communes, comme suit :

   ```
   . ../lib/common_function.sh
   ```

3. Ajoutez le code afin d'obtenir la valeur pour la propriété de prérequis en utilisant les commandes et les options spécifiques à cette plate-forme pour accéder aux éléments de l'environnement hôte. Par exemple, la`env.path.jar` propriété de prérequis doit vérifier si JRE est défini dans`PATH` la variable. Le code suivant exécute la `env` commande, recherche la sortie pour le chemin de `PATH` la variable, puis recherche sa valeur pour le chemin d'accès au JRE.

   ```
   envJRE=`env | grep "PATH" | grep -w "/opt/IBM/Java60/jre/bin"`
   ```

4. Assurez-vous que la vérification renvoie la sortie standard:

   ```
   Répercutez "Vrai"|"Faux" 'si l'analyse vérifie l'existence de la propriété de prérequis
        propriété
   Répercutez $res 'si les vérifications d'analyse renvoient la valeur, par exemple,
   la version du produit, à supprimer de la propriété de prérequis
   Répercutez "Indisponible" 'si l'analyse ne renvoie pas de valeur pour la propriété de prérequis
   Répercutez "Disponible" 'l'analyse renvoie une vérification valide pour la propriété de prérequis
   ```

   Dans l'exemple, basée sur la valeur de `$envJRE` la variable, la vérification renvoie à `True` `False`:

   ```
   if [ $envJRE ]; then
    echo "True"
   else
    echo "False"
   fi
   ```

5. Exécutez le collecteur personnalisé afin de s'assurer qu'il n'existe pas d'erreurs d'exécution et de débogage si nécessaire.
6. Editez le *ips_root*`/UNIX_Linux/script de test de package.sh` pour appeler et exécuter le collecteur personnalisé.
7. Créez un évaluateur personnalisé uniquement si le collecteur personnalisé renvoie des valeurs autres que des valeurs booléennes.

Éditer un script de test de package pour les systèmes UNIX

Vous pouvez mettre à jour le fichier de script`packageTest.sh` pour lancer des collecteurs personnalisés sur les systèmes UNIX.

Avant de commencer

Assurez-vous que vous connaissez les noms des collecteurs associés avec des propriétés de prérequis comme indiqué dans Annexe D, «Collecteurs prédéfinis pour systèmes UNIX», à la page 117. Si la propriété de prérequis est subdivisée en application, en fonctionnalité ou en sous-type de service transférez le différenciateur pour le sous-type *suffix_identifier*, c'est-à-dire, *differentiator_suffix_identifier* vers son collecteur.

Par exemple, `os.package` est le collecteur commun qui permet de vérifier les packages existants. Pour vérifier l'existence de`openssh`, transférez le nom du package en lançant le collecteur `os.package` dans le fichier de script`packageTest.sh` comme suit :

```
./os.package openssh
```

Où `openssh` est le nom du package, c'est-à-dire, le sous-type *suffix_identifier* et le différenciateur *differentiator_suffix_identifier*.

Procédure

1. A l'aide d'un éditeur, ouvrez le script *ips_root*/UNIX_Linux/packageTest.sh.
2. Ajoutez le code pour lire la propriété personnalisée depuis le fichier de configuration et évaluez sa valeur.

   ```
   res=`echo $line | grep [prefix_identifier.]property_name[.suffix_identifier]`
   S'il s'agit de [ $res ]; puis
   ExpValue=`echo $res | cut -d "=" -f2`
   ```

   Par exemple, pour lire la propriété de prérequis personnalisée env.path.jar et vérifier si l'environnement d'exécution Java est défini dans la variable PATH :

   ```
   res=`echo $line | grep env.path.jar`
   if [ $res ]; then
   ExpValue=`echo $res | cut -d "=" -f2`
   ```

   Dans l'exemple :

   ```
    echo "\`wrlTrace "Starting" "env.path.jar"\`"  >>/tmp/prs.check
    echo "\`wrlTrace "Executing" "env.path.jar"\`"  >>/tmp/prs.check
    echo "\`wrlDebug "Starting" "env.path.jar"\`" >>/tmp/prs.check
    echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check
   ```

3. Lancez des fonctions de logging pour des données de débogage et de suivi avant de lancer le collecteur personnalisé.

   ```
    echo "\`wrlTrace "Starting" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`"  >>/tmp/prs.check
    echo "\`wrlTrace "Executing" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`"  >>/tmp/prs.check
    echo "\`wrlDebug "Starting" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`" >>/tmp/prs.check
    echo "\`wrlDebug "Expected" "ExpValue" \`" >>/tmp/prs.check
   ```

4. Lancez le collecteur personnalisé.

   Remarque : Si le collecteur personnalisé a des sous-types, c'est-à-dire, [*suffix_identifier*] dans le nom du fichier et a besoin de vérifications supplémentaires basées sur le sous-type, transférez le différenciateur [*differentiator_suffix_identifier*] pour le sous-type vers le collecteur personnalisé.

   ```
    echo "ss=\`./[prefix_identifier.]property_name[.suffix_identifier]
    [differentiator_suffix_identifier]\`" >>/tmp/prs.check
   ```

   Dans l'exemple :

   ```
    echo "ss=\`./env.path.jar\`" >>/tmp/prs.check
   ```

   Remarque : Exemples de différenciateurs pour le sous-type *script_name* pour les propriétés de prérequis os.file.*script_name* représentent les chemins d'accès des scripts qui sont transférés vers les collecteurs os.filepath :

   ```
   echo "ss=\`./os.filepath /usr/bin/expect\`" >>/tmp/prs.check #os.file.expect
   echo "ss=\`./os.filepath /usr/bin/tar\`" >>/tmp/prs.check #os.file.tar
   echo "ss=\`./os.filepath /usr/bin/gzip\`" >>/tmp/prs.check #os.file.gzip
   ```

5. Lancez des fonctions de logging pour des données de débogage et de suivi lorsque vous quittez le collecteur personnalisé.

   ```
    echo "\`wrlTrace "Finished" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`" >/tmp/prs.check
    echo "echo \"[prefix_identifier.]property_name
   [.suffix_identifier]=\$ss\"" >>/tmp/prs.check
    echo "\`wrlDebug "Finished" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`"  >>/tmp/prs.check
    echo "\`wrlDebug "OutPutValueIs" \$ss\`" >/tmp/prs.check
    echo "\`wrlTrace "Done" "[prefix_identifier.]property_name
   [.suffix_identifier]"\`"  >>/tmp/prs.check
   fi
   ```

Dans l'exemple :

```
 echo "ss=\`./env.path.jar\`" >>/tmp/prs.check
 echo "\`wrlTrace "Finished" "env.path.jar"\`"  >>/tmp/prs.check
 echo "echo \"env.path.jar=\$ss\"" >>/tmp/prs.check
 echo "\`wrlDebug "Finished" "env.path.jar"\`"  >>/tmp/prs.check
 echo "\`wrlDebug "OutPutValueIs" \$ss\`"  >>/tmp/prs.check
 echo "\`wrlTrace "Done" "env.path.jar"\`"  >>/tmp/prs.check
fi
```

6. Répétez les étapes 2 à 5 pour chaque propriété de prérequis personnalisée.

Créer des évaluateurs personnalisés pour les systèmes Windows

Vous pouvez créer des évaluateurs VBScript si les évaluateurs de base ne comparent pas les valeurs réelles et escomptées pour les propriétés de prérequis en utilisant les bons critères d'évaluation. Lorsque vous créez des évaluateurs personnalisés, le nom du fichier doit se terminer par `_compare` et être stocké dans le sous-répertoire `/Windows`. L'évaluateur personnalisé peut utiliser des fonctions communes et des sous-routines pour comparer les valeurs si nécessaire.

Avant de commencer

Assurez-vous de revoir l'ensemble des fonctions et sous-routines dans les annexes suivantes avant de créer l'évaluateur. Déterminez si vous pouvez utiliser n'importe quelle d'entre elles pour comparer les valeurs :

- Annexe E, «Fonctions communes pour les systèmes Windows», à la page 123
- Annexe G, «Utilitaire de fichiers sous-routines pour les systèmes Windows», à la page 139
- Annexe F, «Sous-routines des utilitaires de journalisation pour les systèmes Windows», à la page 137
- Annexe H, «Autres fonctions communes et sous-routines pour les systèmes Windows», à la page 141

Remarque : La fonction commune «succès ou échec ()», à la page 133, peut comparer les valeurs réelles et escomptées pour les types de données suivants : un nombre générique ; taille en Mo et en Go ; vitesse de processeur en MHz ou en GHz; valeur booléenne ; ou une chaîne. Créez uniquement un évaluateur personnalisé si la fonction`passOrFail` est inutilisable.

Procédure

1. Créez un fichier VBScript. Enregistrez le fichier dans le répertoire *ips_root*`/Windows`, avec une variante de la convention d'attribution de noms :

   `[`*prefix_identifier*`.]`*property_name*`[.`*suffix_identifier*`]_compare.vbs`

   où :

   - *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis comme indiqué dans tableau 3, à la page 4. Cet identificateur préfixe est requis par quelques-unes des catégories prédéfinies.
   - *property_name* est le nom de la propriété de prérequis.
   - *suffix_identifier* est un identificateur optionnel pour un sous-type de propriétés de prérequis comme indiqué dans tableau 4, à la page 7.

2. Ajoutez le code pour comparer les valeurs réelles et escomptées qui sont transmises à l'évaluateur comme argument et fonctions associées. Assurez-vous que la comparaison retourne la sortie standard comme suit :
   - "PASS" lorsque la valeur escomptée pour la propriété de prérequis est supérieure ou égale à la valeur réelle de la propriété de prérequis.
   - "FAIL" lorsque la valeur escomptée pour la propriété de prérequis n'est pas égale à la valeur réelle de la propriété de prérequis
3. Lancez l'évaluateur, si nécessaire, pour s'assurer qu'il n'y a pas d'erreurs d'exécution et de débogage.

Exemple

Cet évaluateur personnalisé vérifie les valeurs réelles et escomptées pour la version de Tivoli Directory Integrator. Il utilise la fonction commune, «versionCompare()», à la page 143.

```
wscript.echo "expect: " &#38; wscript.arguments(0)
wscript.echo "real value: " &#38; wscript.arguments(1)
wscript.echo tdiVersionCompare(wscript.arguments(0), wscript.arguments(1))

function tdiVersionCompare(expect, real)
    if len(real) = 0 then
        tdiVersionCompare = "FAIL"
        exit function
    end if

    expect = Trim(expect)
    real = Trim(real)

    Dim expectedVersion
    'if (StrComp(Right(expect,1),"+")=0 or StrComp(Right(expect,1),"-")=0) Then
    if (Right(expect,1)="+" or Right(expect,1)="-") Then
        expectedVersion = Left(expect,len(expect)-1)
    else
        expectedVersion = expect
    end if

    Dim cmp
    cmp = versionCompare(expectedVersion,real)

    if (StrComp(Right(expect,1),"+")=0) Then
        ' La version doit être au moins une valeur escomptée
        if (cmp=0 or cmp=-1) Then
            tdiVersionCompare = "PASS"
        else
            tdiVersionCompare = "FAIL"
        end if
    elseif (StrComp(Right(expect,1),"-")=0) Then
            'La version doit être inférieure ou égale à la valeur escomptée
            if (cmp=0 or cmp=1) Then
                tdiVersionCompare = "PASS"
            else
                tdiVersionCompare = "FAIL"
            end if
    elseif cmp=0 then
        tdiVersionCompare = "PASS"
    else
        tdiVersionCompare = "FAIL"
    end if
end function

' Fonction générique pour comparer deux chaînes de version
'
' Paramètres
```

```
'       ver1 La première chaîne de version
'       ver2 La deuxième chaîne de version
'
' Les chaînes de version ver1 et ver2 doivent être séparées par un point
'Les chaînes de version (ex: 1.0.0.4, 2.3, 3.40.26.7800, 2.3.a) peuvent avoir tous les
' numéros de composants. Lorsque vous comparez plusieurs numéros
' de composants, les composants manquants  de la plus courte chaîne de version seront traités
' comme s'il y avait un zéro là-bas. Si aucun caractère non numérique n'est
' inclus dans un composant de version, ces composants correspondants seront comparés
' comme des chaînes et ne seront pas analysés sous forme numérique
'
' Returns
'       1 version1 > version2
'      -1 version1 &#60; version2
'       0 version1 = version2
'
' Cas spéciaux :
' RESULT    version 1    version 2
'   0         empty        empty
'   1      validString     empty
'  -1         empty     validString
'
' Remarque : Cette fonction devrait éventuellement être déplacée vers common_functions.vbs

function versionCompare(ver1, ver2)
    WScript.echo "Comparing [" &#38; ver1 &#38; "] to [" &#38; ver2 &#38; "]"

    Const UNASSIGNED = "*UNASSIGNED*"
    Dim v1Default, v2Default

    ' Traiter des cas spéciaux :
    if (IsEmpty(ver1) and IsEmpty(ver2)) Then
        versionCompare = 0
        exit function
    end if
    if (IsEmpty(ver1) and not IsEmpty(ver2)) Then
        versionCompare = -1
        exit function
    end if
    if (not IsEmpty(ver1) and IsEmpty(ver2)) Then
        versionCompare = 1
        exit function
    end if

    Dim ver1Parts, ver2Parts

    'Les versions ne sont pas vides.  Fractionner les composants et comparez les nombres
    ver1Parts = Split(ver1,".")
    ver2Parts = Split(ver2,".")

    Dim v1Size, v2Size
    v1Size = ubound(ver1Parts)
    v2Size = ubound(ver2Parts)

    ' If last version part is "*", treat all missing parts as "*"
    '(so 2.* matches 2.1.3, for example)
    if (v1Size > v2Size) Then
        Redim Preserve ver2Parts(v1Size)
        if (ver2Parts(v2Size)="*") Then
            for i = v2Size to v1Size
                ver2Parts(i) = "*"
            next
        end if
    elseif (v2Size > v1Size) Then
        Redim Preserve ver1Parts(v2Size)
        if (ver1Parts(v1Size)="*") Then
            for i = v1Size to v2Size
```

```
            ver1Parts(i) = "*"
        next
    end if
end if

Dim i
i = 0

Do While (i&#60;=ubound(ver1Parts) or i&#60;=ubound(ver2Parts))
    Dim v1, v2, v1Str, v2Str

    v1Str = UNASSIGNED
    v2Str = UNASSIGNED

    if (i&#60;=ubound(ver1Parts)) Then
        on error resume next
        v1 = Int(ver1Parts(i))
        if not Err=0 Then
            v1Str = ver1Parts(i)
            if (i&#60;=ubound(ver2Parts)) Then
                v2Str = ver2Parts(i)
            else
                v2Str = "0"
            end if
        end if
    else
        v1 = 0
    end if

    if (i&#60;=ubound(ver2Parts)) Then
        on error resume next
        v2 = Int(ver2Parts(i))
        if not Err=0 Then
            if (i&#60;=ubound(ver1Parts)) Then
                v1Str = ver1Parts(i)
            else
                v1Str = "0"
            end if
            v2Str = ver2Parts(i)
        end if
    else
        v2 = 0
    end if

    if (not v1Str=UNASSIGNED or not v2Str=UNASSIGNED) Then
        if (IsEmpty(v1Str)) Then
            v1Str = "0"
        end if
        if (IsEmpty(v2Str)) Then
            v2Str = "0"
        End if

        'WScript.echo "Comparing as strings: " &#38; v1Str &#38; " : " &#38; v2Str
        ' Compare as Strings if either part could not be converted to a number
        if (not v1Str="*" and not v2Str="*") Then
            if (not v1Str=v2Str) Then
                versionCompare = StrComp(v1Str,v2Str)
                exit function
            end if
        end if
    else
        'WScript.echo "Comparing as numbers: " &#38; v1 &#38; " : " &#38; v2

        if (v1 > v2) Then
            versionCompare = 1
            exit function
        end if
```

```
            if (v2 > v1) Then
                versionCompare = -1
                exit function
            end if
        end if

        i = i + 1
    Loop

    ' Si nous sommes arrivés à ce niveau, les versions doivent être égales
    versionCompare = 0

Terminer la fonction
```

Création d'évaluateurs personnalisés pour les systèmes UNIX

Vous pouvez créer des évaluateurs personnalisées si le collecteur personnalisé ne renvoie pas les valeurs booléennes, c'est-à-dire, `True` ou`False`. Lorsque vous créez des évaluateurs personnalisés, le nom du fichier doit se terminer par `_compare` et être stocké dans le sous-répertoire `/UNIX_Linux`. L'évaluateur personnalisé peut utiliser des fonctions communes pour comparer les valeurs si nécessaire.

Avant de commencer

Assurez-vous de revoir l'ensemble des fonctions dans les annexes suivantes avant de créer les évaluateurs personnalisés. Déterminez si vous pouvez utiliser n'importe quelle d'entre elles pour comparer les valeurs réelles et escomptées :

- Annexe I, «Fonctions communes pour les systèmes UNIX», à la page 145
- Annexe J, «Autres fonctions pour les systèmes UNIX», à la page 153
- Annexe K, «Fonctions de l'utilitaire de journalisation pour les systèmes UNIX», à la page 161

Il y a deux fichiers de script que vous pouvez utiliser comme point de départ : `._compare.sh` et`_compare.sh` dans le sous-répertoire`/Unix_Linux`.

Important : Ne créez pas d'évaluateurs personnalisés si votre évaluateur personnalisé renvoie `True` ou`False`. IBM Prerequisite Scanner utilise des évaluateurs prédéfinis pour tout collecteur qui renvoie des valeurs booléennes.

Procédure

1. Créer un fichier interpréteur de commande. Enregistrez le fichier dans le répertoire*ips_root*`/UNIX_Linux`, avec une variante de convention d'attribution de noms :

   `[`*`prefix_identifier`*`.]`*`property_name`*`[.`*`suffix_identifier`*`]_compare.sh`

   où :

   - *prefix_identifier* est un identificateur pour une catégorie prédéfinie de propriétés de prérequis comme indiqué dans tableau 3, à la page 4. Cet identificateur préfix est requis par quelques unes des catégories prédéfinies.
   - *property_name* est le nom de la propriété de prérequis.
   - *suffix_identifier* est un identificateur optionnel pour un sous-type de propriétés de prérequis comme indiqué dans tableau 4, à la page 7.
2. Ajoutez le code pour comparer les valeurs réelles et escomptées qui sont transmises à l'évaluateur en tant qu'argument et fonctions associées. Assurez-vous que la comparaison retourne la sortie standard comme suit :

- "PASS" lorsque la valeur escomptée pour la propriété de prérequis est supérieure ou égale à la valeur réelle de la propriété de prérequis.
- "FAIL" lorsque la valeur escomptée pour la propriété de prérequis n'est pas égale à valeur réelle de la propriété de prérequis

3. Lancez l'évaluateur, si nécessaire, pour s'assurer qu'il n'y a pas d'erreurs d'exécution et de débogage.

Chapitre 4. Exécution de Prerequisite Scanner

Vous pouvez utiliser l'interface de ligne de commande pour exécuter IBM Prerequisite Scanner. Le script Prerequisite Scanner `prereq_checker` utilise un ensemble de paramètres obligatoires et facultatifs ainsi qu'un indicateur de commande pour les paramètres facultatifs supplémentaires.

tableau 12 présente les caractères spéciaux utilisés dans la syntaxe du script Prerequisite Scanner.

Tableau 12. Légende des caractères spéciaux pour le script Prerequisite Scanner

Caractères spéciaux	Description
<>	Identifie un nom de marque de réservation.
[]	Identifie un paramètre facultatif. Les paramètres ne doivent pas être notés entre crochets.
...	Indique vous pouvez spécifier plusieurs valeurs pour un paramètre.
|	Indique les paramètres s'excluant mutuellement. Spécifiez le paramètre à gauche du séparateur ou celui à droite du séparateur, mais pas les deux.
{}	Comprend un ensemble de paramètres s'excluant mutuellement séparés par |.

prereq_checker

Le script `prereq_checker` exécute IBM Prerequisite Scanner et vérifie les prérequis en fonction du jeu de paramètres que vous spécifiez lorsque vous exécutez le script.

Syntaxe

```
prereq_checker.bat|sh
  "Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
  [detail]
  [outputDir="ips_output_dir"]
  [xmlResult]
  [PATH="product_root"]
  [-p Product_Code.instance.parameter=value,...]
  [debug]
  [trace]
```

Le script `prereq_checker` dispose d'un seul paramètre obligatoire et de plusieurs paramètres facultatifs.

«"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."», à la page 68
Paramètre obligatoire

«[detail]», à la page 68
Paramètre facultatif

«[outputDir="ips_output_dir"]», à la page 71
Paramètre facultatif

«[xmlResult]», à la page 71
Paramètre facultatif

«[PATH="product_root"]», à la page 71

«[-p Product_Code.instance.parameter=value,...]», à la page 72
Indicateur facultatif

«[debug]», à la page 72
Paramètre facultatif

«[trace]», à la page 72
Paramètre facultatif

"*Product_Code* [*Product_Version*][,*Product_CodeN* [*Product_VerN*]]..."

Vous devez définir au moins un paramètre **Product_Code** pour identifier le produit ou le composant pour lequel vous exécutez la vérification des prérequis et le fichier de configuration connexe. **Product_Code** est le code produit que vous définissez dans le fichier *ips_root*/`codename.cfg`.

Par exemple, `KMS` est le code produit de Tivoli Enterprise Monitoring Server dans le fichier `product.cfg`. Pour exécuter le scanner, entrez le script suivant avec le code produit :

```
./prereq_checker.sh KMS
```

Si vous définissez un paramètre **Product_Code** qui ne dispose pas d'un fichier de configuration approprié, Prerequisite Scanner l'ignore sans erreur. Le fichier journal contient un message indiquant qu'aucun fichier de configuration n'a été trouvé.

Le paramètre **Product_Version** paramètre associé **Product_Code** indique la version du produit. Il s'agit du code à 8 chiffres représentant la version, l'édition, la modification et le niveau, avec deux chiffres pour chaque partie du code, par exemple, 7.3.21 désigne 07032100. **Product_Version** est un paramètre facultatif. Si vous ne l'avez pas défini, Prerequisite Scanner vérifie la dernière version disponible.

Vous pouvez définir un ou plusieurs paramètres **Product_Code** avec le paramètre facultatif **Product_Version**, séparé les uns des autres par une virgule.

Important : Lorsque vous définissez plusieurs paramètres **Product_Code** avec le paramètre facultatif **Product_Version**, mettez-les entre guillemets. Sinon, le scanner échoue.

Cet exemple vérifie les prérequis de la dernière version de Tivoli Monitoring Operating System Agent for Windows et de la version 6.2.1 de Tivoli Monitoring Agent for DB2.

```
prereq_checker.bat "KNT,KUD 06210000"
```

[detail]

Ce paramètre facultatif indique s'il faut afficher les résultats détaillés de l'analyse dans l'interface de ligne de commande.

Important : Ne mettez pas ce paramètre entre guillemets.

Lorsque vous définissez le paramètre **detail**, les résultats détaillés contiennent :

- La version de Prerequisite Scanner
- La version du système d'exploitation sur lequel le scanner a été exécuté

- Le nom des produits ou des composants pour lesquels les contrôles prérequis ont été exécutés
- Pour chaque propriété de prérequis : le nom de la propriété de prérequis contrôlée, le résultat `PASS` ou `FAIL`, la valeur réelle et la valeur attendue
- Pour tous les composants : le nom de la propriété de prérequis générale contrôlée, le résultat `PASS` ou `FAIL`, la valeur réelle et la valeur attendue
- Le résultat global `PASS` ou `FAIL`

Prerequisite Scanner enregistre également ces résultats dans le fichier *ips_output_dir*/`result.txt`. Il enregistre les résultats dans le fichier texte indépendamment du fait que vous définissez le paramètre **`detail`**.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
[root@aclinux15 20110927-0849]# ./prereq_checker.sh DMO detail
IBM Prerequisite Scanner
     Version: 1.1.1.8
     Build  : 20110927
     OS Name: Linux

 Machine Info
 Machine Name : <Machine name>
 Serial Number: <Serial number>

TPS detected : Red Hat Enterprise Linux Server release 5.5 {32-bit}
Using the DMO config file
Using config file - /root/prs/20110927-0849/UNIX_Linux/DMO_0750000.cfg for DMO
DMO - Prerequisite Scanner Demo [0750000]:
Evaluation                          PASS/FAIL Result                                Expected Result

DBType                              FAIL      Unknown                               Oracle

DBType                              FAIL      Unknown                               DB2

DBType                              FAIL      Unknown                               regex{.*Oracle.
*}
DBType                              FAIL      Unknown                               regex{.*DB2.*}

DBTypeDetails                       FAIL      Unknown                               oracle

DBTypeDetails                       FAIL      Unknown                               DB2

DBTypeDetails                       FAIL      Unknown                               regex{.*Oracle.
*}
DBTypeDetails                       FAIL      Unknown                               regex{.*DB2.*}

OS Version                          PASS      "Red Hat Enterprise Linux Server release 5.5 (Tikanga)"
 " regex{Red Hat.*Tikanga.*}"
                                                                                    regex{AIX.*}

                                                                                    regex{Solaris.*
}
os.lib.libstdc++                    PASS      /usr/lib/gcc/i386-redhat-linux/4.1.1/libstdc++.so libst
dc++
os.lib.libgcc                       PASS      /usr/lib/gcc/i386-redhat-linux/3.4.6/libgcc_s.so [Check
Package:True]regex{libgcc.*}
os.lib.libXp                        PASS      /usr/lib/libXmu.so.6                  regex{libX.*}

os.space.var                        PASS      "38GB"                                " [dir:root=/va
r/ibm/common/acsi"
                                                                                    unit:MB]1.0

os.space.usr                        PASS      "38GB"                                " [dir:root=/us
r/ibm/common/acsi"
                                                                                    unit:MB]200

os.space.tmp                        PASS      36GB                                  30MB

env.classpath.derbyJAR              PASS      False                                 False

network.pingSelf                    PASS      True                                  True

env.classpath.derbyJAR              PASS      False                                 False

network.pingLocalhost               PASS      True                                  True

os.package.compat-libstdc++-33      PASS      compat-libstdc++-33-3.2.3-61          compat-libstdc+
+-33

TOTAL ALL SPECIFIED COMPONENTS:
Evaluation                          PASS/FAIL Result                                Expected Result

/                                   PASS      38.00GB                               201MB

/tmp                                PASS      36.00GB                               30MB


Prereq Check  Overall Result:   FAIL
[root@aclinux15 20110927-0849]#
```

Figure 10. Exécution du script et définition du paramètre de détails sur les systèmes UNIX

Si vous ne définissez pas le paramètre **detail**, le scanner affiche uniquement le résultat PASS ou FAIL dans l'interface de ligne de commande.

Figure 11. Exécution du script sans la définition du paramètre de détails sur les systèmes Windows

[outputDir="*ips_output_dir*"]

Ce paramètre facultatif indique que vous souhaitez définir le répertoire de sortie des résultats de l'analyse et les fichiers journaux pour Prerequisite Scanner.

Lorsque vous exécutez le script Prerequisite Scanner et définissez le paramètre **outputDir** facultatif, Prerequisite Scanner affiche le texte des résultats , XML et les fichiers journaux dans le répertoire spécifié par la valeur du paramètre. Cette valeur est connue sous le nom de *ips_output_dir* dans la documentation.

Si vous ne définissez pas ce paramètre, l'emplacement de sortie par défaut est *ips_root*.

Vous devez utiliser le paramètre pour indiquer un emplacement, si vous choisissez d'exécuter Prerequisite Scanner à partir d'un CD, d'un DVD ou d'une unité réseau en lecture seule. Vous devez disposer des permissions d'écriture sur *ips_output_dir*; sinon, Prerequisite Scanner échoue.

Important : Si le répertoire de sortie n'existe pas, Prerequisite Scanner crée le répertoire. Vous devez disposer des permissions d'écriture pour créer ou écrire dans le répertoire de sortie dans lequel Prerequisite Scanner enregistre les fichiers.

[xmlResult]

Ce paramètre facultatif indique que vous souhaitez afficher les résultats dans le fichier des résultats XML et dans le fichier des résultats de test clair.

Lorsque vous exécutez le script Prerequisite Scanner et définissez le paramètre **xmlResult** facultatif, Prerequisite Scanner affiche les résultats dans le fichier *ips_output_dir*`/result.xml`.

Si vous ne définissez pas ce paramètre, les résultats sont affichés uniquement dans le fichier en texte clair.

[PATH="*product_root*"]

Ce paramètre facultatif indique les répertoires d'installation des produits.

Important : Sous Windows, ne définissez pas le chemin d'accès à un identificateur d'unité uniquement, c'est-à-dire `C:`. Assurez-vous que vous définissez un chemin d'accès valide.

Si vous ne définissez pas le paramètre **path**, le scanner affiche les répertoires d'installation par défaut des produits IBM Tivoli.

- Sous les systèmes **UNIX :** `/opt/ibm/itm`
- Sous les systèmes **Windows :** `C:\IBM\itm`

[-p *Product_Code.instance.parameter=value,...*]

L'indicateur facultatif **-p** indique que les paramètres de la procédure doit être transmis à un fichier de script pour un contrôle prérequis supplémentaire. **<Product_Code>** désigne le code produit. Seul chaque jeu de *instance.parameter=value* est transmis au script. Vous pouvez transmettre plusieurs jeux de paramètres, séparés par une virgule.

Le script auquel les paramètres sont transmis est déterminé par les options suivantes :

- Avec un préfixe **Product_Code**, les paramètres sont transmis au script à l'aide du **Product_Code** connexe
- Sans le préfixe **Product_Code**, les paramètres sont transmis au collecteurs communs.

Exemple 1`-p KUD.inst1.DB2_INST_OWNER=db2inst1, KUD.inst2.DB2_INST_OWNER=db2inst2` Cet indicateur avec des paramètres transmet `db2inst1.DB2_INST_OWNER=db2inst1` et `db2inst2.DB2_INST_OWNER=db2inst2` au fichier script `KUD.`**`Product_Version`**`.bat`.

Exemple 2

```
-p SERVER=IP.PIPE://mymachine:1918
```

Cet indicateur avec des paramètres transmet `SERVER=IP.PIPE://mymachine:1918` au collecteur commun pour vérifier les ports.

Remarque : Ce script accepte les paramètres dans **-p** en tant que `tacmd createNode`.
Vous pouvez définir les paramètres `SERVER`, `PROTOCOL`, `PORT`, `BACKUP` et `BSERVER` dans *ips_root*`/lib/common_configuration`. Prerequisite Scanner définit les priorités des paramètres transmis à partir de l'interface de ligne de commande avant ceux du fichier `common_configuration`.

[debug]

Ce paramètre facultatif indique que vous souhaitez mettre sous tension le débogage tout en exécutant Prerequisite Scanner.

Lorsque vous exécutez le script Prerequisite Scanner et définissez le paramètre **debug** facultatif, Prerequisite Scanner génère des informations de traitement détaillées, des messages d'avertissement et d'erreur. Les résultats de l'analyse sont affichés dans le fichier journal. Il s'agit du fichier *ips_output_dir*`/prs.debug` sous les systèmes UNIX et du fichier *ips_output_dir*`/precheck.log` sous les systèmes Windows.

Important : Le débogage du scanner est désactivé par défaut.

[trace]

(Systèmes UNIX uniquement) Ce paramètre facultatif indique que vous souhaitez mettre sous tension la consignation de trace tout en exécutant Prerequisite Scanner.

Lorsque vous exécutez le script Prerequisite Scanner et définissez le paramètre **trace** facultatif, Prerequisite Scanner génèrent les informations de trace dans le fichier *ips_output_dir*`/prs.trc`.

Important : L'historique de trace du scanner est désactivé par défaut.

Exécution de Prerequisite Scanner à partir de la ligne de commande

Vous pouvez exécuter IBM Prerequisite Scanner à partir d'une interface de ligne de commande et saisir les paramètres d'entrée associés pour le script.

Avant de commencer

Recherchez dans la documentation d'installation ou les notes techniques de votre produit les étapes supplémentaires qui doivent être exécutées avant d'exécuter Prerequisite Scanner. Par exemple, vous pouvez définir la variable d'environnement indiquant à Prerequisite Scanner les composants ou fonctionnalités installé(e)s sur l'ordinateur cible et, par conséquent, les conditions prérequises à contrôler.

Procédure

1. Ouvrez l'interface de ligne de commande et le répertoire *ips_root*.
2. exécutez le fichier script Prerequisite Scanner **prereq_checker** comme suit :

   UNIX

   ```
   ./prereq_checker.sh
   "Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
   [detail]
   [outputDir="ips_output_dir"]
   [xmlResult]
   [PATH="product_root"]
   [-p Product_Code.instance.parameter=value,...]
   ```

   Dans l'exemple ci-dessous, le système exécute Prerequisite Scanner for Autonomic Deployment Engine à l'aide d'un fichier de configuration et de son code produit associé `ADE`:

   ```
   ./prereq_checker.sh
   ADE 072000
   detail
   PATH=/opt/ibm/tivoli
   ```

   Windows

   ```
   prereq_checker.bat
   "Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."
   [detail]
   [outputDir="ips_output_dir"]
   [xmlResult]
   [PATH="product_root"]
   [-p Product_Code.instance.parameter=value,...]
   ```

   Dans l'exemple ci-dessous, le système exécute Prerequisite Scanner for Tivoli Provisioning Manager for Windows 2003 et 2008 à l'aide des codes produit `COX` et `COY`.

   ```
   prereq_checker.bat
   "COX, COY 07200000"
   detail
   PATH="D:\ibm\tivoli"
   -p SERVER=IP.PIPE://mytems:1234
   ```

   Dans l'exemple ci-dessous, le système exécute Prerequisite Scanner for Tivoli zEnterprise Monitoring Agent à l'aide du code produit `KZE`. Il définit également l'emplacement des résultats et des fichiers journaux dans *ips_output_dir* à l'aide du paramètre **outputDir** facultatif.

Important : Vous devez utiliser le paramètre **outputDir** pour spécifier un emplacement si vous décidez d'exécuter Prerequisite Scanner à partir d'un CD, d'un DVD ou d'une unité réseau en lecture seule. Vous devez disposer des droits en écriture pour *ips_output_dir* ; sinon, Prerequisite Scanner échoue.

```
Windows
prereq_checker.bat
"KZE 06230000"
outputDir="%TEMP%\ips"
UNIX
./prereq_checker.sh
"KZE 06230000"
outputDir="/tmp/ips"
```

Le scanner affiche les fichiers `result.txt file` et `precheck.log` dans les emplacements suivants :

- Sous les systèmes Windows : `D:\temp\ips` où `TEMP` est la variable d'environnement du dossier temporaire.
- Sur les systèmes UNIX : `/tmp/ips`

Important : Si le répertoire de sortie n'existe pas, Prerequisite Scanner le crée. Vous devez disposer des droits en écriture pour créer ou écrire dans le répertoire de sortie dans lequel Prerequisite Scanner enregistre les fichiers.

Emplacements de répertoire commun

Il existe des variables de nom de chemin pour les répertoires communs.

Répertoire d'installation d'IBM Prerequisite Scanner

ips_root décrit l'emplacement où Prerequisite Scanner est installé. Cet emplacement peut être spécifié lors de l'installation.

Répertoire de sortie Prerequisite Scanner

ips_output_dir décrit l'emplacement où les résultats de l'analyse et les fichiers de Prerequisite Scanner sont sauvegardés. Cet emplacement peut être spécifié à l'aide du paramètre d'entrée **outputDir** lorsque vous exécutez le Scanner. Si vous ne définissez pas ce paramètre, l'emplacement de sortie par défaut est *ips_root*.

Remarque : Prerequisite Scanner crée des fichiers temporaires lors de son exécution, mais ces fichiers sont supprimés avant que le scanner ne termine son exécution. Ces fichiers temporaires se trouvent dans le sous-répertoire *ips_output_dir*/`temp`. Le Scanner supprime également le sous-répertoire *ips_output_dir*/`temp`, à moins que ce dernier ne contienne des fichiers de débogage et de trace qui sont générés sur les systèmes UNIX uniquement.

Chapitre 5. Identification et résolution des problèmes de Prerequisite Scanner

Vous pouvez identifier et résoudre les problèmes de IBM Prerequisite Scanner à l'aide des fichiers journaux et des fonctions de consignation lorsque vous créez des contrôles de prérequis personnalisés.

Prerequisite Scanner génère des codes de retour en fonction des résultats de l'analyse et du fait qu'il faille éventuellement quitter le système en raison d'erreurs. Ces codes de retour sont écrits dans les fichiers journaux. Par exemple, si Prerequisite Scanner ne parvient pas à exécuter l'analyse car il ne peut pas lire le fichier de configuration, il génère le code de retour de `2`.

Identification et résolution des problèmes sous les systèmes Windows

Lorsque vous exécutez IBM Prerequisite Scanner, le système crée un fichier journal par défaut. Il contient des informations détaillées avec chaque étape et chaque fonction que le scanner exécute en séquence. Le fichier contient également des horodatages comprenant les heures de début et fin de chaque fonction et chaque étape. Vous pouvez déboguer et modifier le fichier journal pour déterminer où et quand l'erreur s'est produite.

Prerequisite Scanner affiche les informations de traitement, les messages d'avertissement et d'erreur ainsi que les résultats d'analyse dans le fichier *ips_output_dir*`/precheck.log`. Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **`debug`**, Prerequisite Scanner affiche les messages de débogage supplémentaires dans ce fichier.

figure 12 présente un exemple de fichier journal lorsque le paramètre facultatif **debug** est défini et figure 13, à la page 77 affiche le fichier journal lorsque le paramètre n'est pas défini.

```
precheck.log - Notepad
File  Edit  Format  View  Help
[9/27/2011 1:27:34 PM] INFO   :
IBM Prerequisite Scanner
   Version  : 1.1.1.8
   Build    : 20110927
   OS Name  : Microsoft Windows XP Professional Service Pack 3
   User Name: <User Name>
[9/27/2011 1:27:34 PM] INFO   :
Machine Info
  Machine name : <Machine name>
  Serial Number: <Serial number>
  OS Serial    : <OS serial number>

[9/27/2011 1:27:34 PM] INFO   : Parsing command-line arguments
[9/27/2011 1:27:34 PM] INFO   : arguments(0) = DMO
[9/27/2011 1:27:34 PM] INFO   : arguments(1) = detail
[9/27/2011 1:27:34 PM] INFO   : arguments(2) = debug
[9/27/2011 1:27:34 PM] INFO   : codesAndVersions = DMO
[9/27/2011 1:27:34 PM] INFO   : No PATH paramter specified. The default value of C:\ibm\ITM will be used
[9/27/2011 1:27:34 PM] DEBUG  : Detected operating system types are [Windows|Windows Workstation|Windows XP]
[9/27/2011 1:27:34 PM] DEBUG  : Detected operating system version [5.1.2600]
[9/27/2011 1:27:34 PM] DEBUG  : Detected service pack level [3.0]
[9/27/2011 1:27:34 PM] INFO   : Step 1 - Scanning product codes
[9/27/2011 1:27:34 PM] INFO   : Input PD codes and versions = [DMO]
[9/27/2011 1:27:34 PM] INFO   : Finding available configuration files in [C:\prs\precheck_windows_20110927\Windows]
[9/27/2011 1:27:34 PM] INFO   : config file mapping = Dictionary[
        DMO=Dictionary[
        cfg=DMO_01000000.cfg
        exe=common.bat
]
]
[9/27/2011 1:27:34 PM] INFO   : Step 1.5 - Adding default parameters if necessary
[9/27/2011 1:27:34 PM] INFO   : deliverPara=[]
[9/27/2011 1:27:34 PM] INFO   : DMO will read: DMO_01000000.cfg and will execute script: common.bat
[9/27/2011 1:27:34 PM] INFO   : Step 2 - Collect properties
[9/27/2011 1:27:34 PM] INFO   : start commands: [ 1 scripts will be called ]
[9/27/2011 1:27:34 PM] DEBUG  : The config file is: C:\prs\precheck_windows_20110927\Windows\DMO_01000000.cfg
[9/27/2011 1:27:34 PM] INFO   :  calling common.bat [ 0 scripts left ]
[9/27/2011 1:27:36 PM] INFO   :    [ common.bat ] driveName is [C:]
[9/27/2011 1:27:36 PM] INFO   :    [ common.bat ] Disk is [1413MB]
[9/27/2011 1:27:36 PM] DEBUG  : Processing this line from cfg file: [OS Version=regex{Windows .*}]
[9/27/2011 1:27:36 PM] DEBUG  : Take the first part of the line:   [OS Version]
[9/27/2011 1:27:36 PM] DEBUG  : See if a corresponding vbs or bat file exists for [OS Version]
[9/27/2011 1:27:36 PM] DEBUG  : See if a corresponding vbs or bat file exists for [OS Versio]
```

Figure 12. Fichier precheck.log avec les données de débogage

```
precheck.log - Notepad
File  Edit  Format  View  Help
[9/27/2011 1:25:01 PM] INFO   :
IBM Prerequisite Scanner
   Version  : 1.1.1.8
   Build    : 20110927
   OS Name  : Microsoft Windows XP Professional Service Pack 3
   User Name: <User Name>
[9/27/2011 1:25:01 PM] INFO   :
Machine Info
  Machine name : <Machine name>
  Serial Number: <Serial number>
  OS Serial    : <OS serial number>

[9/27/2011 1:25:01 PM] INFO   : Parsing command-line arguments
[9/27/2011 1:25:01 PM] INFO   : arguments(0) = DMO
[9/27/2011 1:25:01 PM] INFO   : arguments(1) = detail
[9/27/2011 1:25:01 PM] INFO   : codesAndVersions = DMO
[9/27/2011 1:25:01 PM] INFO   : No PATH paramter specified. The default value of C:\ibm\ITM will be used
[9/27/2011 1:25:01 PM] INFO   : Step 1 - Scanning product codes
[9/27/2011 1:25:01 PM] INFO   : Input PD codes and versions = [DMO]
[9/27/2011 1:25:01 PM] INFO   : Finding available configuration files in [C:\prs\precheck_windows_20110927\Windows]
[9/27/2011 1:25:01 PM] INFO   : config file mapping = Dictionary[
        DMO=Dictionary[
        cfg=DMO_01000000.cfg
        exe=common.bat
]
]
[9/27/2011 1:25:01 PM] INFO   : Step 1.5 - Adding default parameters if necessary
[9/27/2011 1:25:01 PM] INFO   : deliverPara=[]
[9/27/2011 1:25:01 PM] INFO   : DMO will read: DMO_01000000.cfg and will execute script: common.bat
[9/27/2011 1:25:01 PM] INFO   : Step 2 - Collect properties
[9/27/2011 1:25:01 PM] INFO   : start commands: [ 1 scripts will be called ]
[9/27/2011 1:25:01 PM] INFO   :   calling common.bat [ 0 scripts left ]
[9/27/2011 1:25:03 PM] INFO   :     [ common.bat ] driveName is [C:]
[9/27/2011 1:25:03 PM] INFO   :     [ common.bat ] Disk is [1413MB]
[9/27/2011 1:25:03 PM] WARNING : 'OS Version' : No supporting vbs or bat file exists for this check!
[9/27/2011 1:25:03 PM] WARNING : 'Memory' : No supporting vbs or bat file exists for this check!
[9/27/2011 1:25:03 PM] INFO   :     [ ] os.versionNumber=5.1.2600
[9/27/2011 1:25:03 PM] INFO   :     [ ] os.servicePack=3.0
[9/27/2011 1:25:03 PM] INFO   :     [ ] os.architecture=32-bit
[9/27/2011 1:25:03 PM] INFO   :     [ ] os.totalPhysicalMemory=3070MB
[9/27/2011 1:25:04 PM] INFO   :     [ ] os.is8dot3FileFormatEnabled=True
[9/27/2011 1:25:04 PM] INFO   :     [ ] os.isServiceRunning.terminalServices=True
[9/27/2011 1:25:04 PM] INFO   :     [ ] os.remoteRegistryEnabled=True
```

Figure 13. Fichier precheck.log sans données de débogage

Identification et résolution des problèmes sur les systèmes UNIX

L'écriture de messages dans les fichiers journaux est désactivée par défaut sur les systèmes UNIX. Vous pouvez activer les fonctions de débogage et de trace à l'aide des paramètres d'entrée **debug** et **trace**. Le scanner écrit les données de débogage et de trace dans différents fichiers journaux et utilise des horodatages pour indiquer les heures de début et de fin des étapes ou des fonctions. Vous pouvez utiliser les deux fichiers pour mettre en corrélation et résoudre une anomalie, une fonction ou un contrôle de prérequis spécifique.

Fichier journal de débogage

Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **debug**, Prerequisite Scanner affiche les informations de traitement détaillées, les messages d'avertissement et d'erreur ainsi que les résultats d'analyse dans le fichier *ips_output_dir*/`temp/prs.debug`. Il contient des informations détaillées avec chaque étape et chaque fonction que le scanner exécute en séquence. Le fichier contient également des horodatages comprenant les heures de début et de fin de chaque fonction et chaque étape. Le sous-répertoire *ips_output_dir*/`temp` contient également les fichiers temporaires `result1.txt` et `result2.txt` qui indiquent l'entrée dans le fichier final *ips_output_dir*/`result.txt`. Vous pouvez utiliser ces fichiers temporaires pour déterminer les anomalies dans les résultats des contrôles de prérequis spécifiques.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
     Build  : 20110927
     OS Name: Linux
[2011.09.27 10.12.15 ] [main()] - Entered
[2011.09.27 10.12.15 ]  ==== Step 1: Detecting OS...
[2011.09.27 10.12.15 ]  OS Detected: Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  product_version: DMO
[2011.09.27 10.12.15 ] [AutoOsDetection()] - Entered
[2011.09.27 10.12.15 ]  [Param] ProductInfo:DMO
[2011.09.27 10.12.15 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  [Param] CPU Arch:Kernel=i686
[2011.09.27 10.12.15 ]  Finding product code in product.cfg
[2011.09.27 10.12.15 ]  product code found :
[2011.09.27 10.12.15 ]  Found DMO code in product.cfg
[2011.09.27 10.12.15 ]  Finding OS Arch and CPU Type
[2011.09.27 10.12.15 ]  Found OS Arch = 32-bit, CPU Type=
[2011.09.27 10.12.15 ]  Calling config_parser.sh...
[2011.09.27 10.12.15 ] [config_parser.sh] - Entered
[2011.09.27 10.12.15 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.15 ]  [Param] ProductCode:DMO
[2011.09.27 10.12.15 ]  [Param] OSArch:Arch=32-bit
[2011.09.27 10.12.16 ]  [Param] CPUArch:CPU=
[2011.09.27 10.12.16 ]  [Param] Version:version=
[2011.09.27 10.12.16 ]  [Param] XXX:Kernel=i686
[2011.09.27 10.12.16 ]  Forming parse array...
[2011.09.27 10.12.16 ] [Form_Parse_String] - Entered
[2011.09.27 10.12.16 ]  [Param] OSInfo:Red Hat Enterprise Linux Server release 5.5 {32-bit}
[2011.09.27 10.12.16 ]  [Param] ProductCode:DMO
[2011.09.27 10.12.16 ]  [Param] OSArch:Arch=32-bit
[2011.09.27 10.12.16 ]  [Param] CPU:CPU=
[2011.09.27 10.12.16 ]  [Param] CPUArch:Kernel=i686
[2011.09.27 10.12.16 ]  Form_Parse_String - ParseArray: [OSType:UNIX][OSType:LINUX][OSType:RedHat][OSType:RedHatEnterpriseLinuxServer][OS
Type:RedHatEnterpriseLinuxServer5.*][OSType:RedHatEnterpriseLinuxServer5.5][OSArch:32-bit][CPUArch:i686]
[2011.09.27 10.12.16 ] [Form_Parse_String] - Exit
[2011.09.27 10.12.16 ]  Reading config file and parsing using parse array...
[2011.09.27 10.12.16 ] [Read_configFile()] - Entered
[2011.09.27 10.12.16 ]  [Param] ConfigFile:/root/prs/20110927-0849/UNIX_Linux/DMO_0750000.cfg-Master
[2011.09.27 10.12.16 ]  [Param] Product:DMO
[2011.09.27 10.12.17 ]  Writing DBType=Oracle to DMO_0750000.cfg
[2011.09.27 10.12.21 ]  Found Env Var - TPAE_DB_Server
```

Figure 14. fichier prs.debug sur les systèmes UNIX

Fichier journal de trace

Lorsque vous exécutez le script Prerequisite Scanner et que vous définissez le paramètre facultatif **trace**, Prerequisite Scanner affiche les informations de trace dans le fichier *ips_output_dir*/temp/prs.trc. Il contient des informations avec chaque fonction que le scanner exécute en séquence. Le fichier contient également des horodatages comprenant les heures de début et de fin de chaque fonction.

```
root@aclinux15:~/prs/20110927-0849
File  Edit  View  Terminal  Tabs  Help
     Build  : 20110927
     OS Name: Linux
[2011.09.27 10.19.58 ] [main()] - Entered:
[2011.09.27 10.19.58 ] [AutoOsDetection()] - Entered:
[2011.09.27 10.19.58 ] [config_parser.sh] - Entered:
[2011.09.27 10.19.59 ] [Form_Parse_String] - Entered:
[2011.09.27 10.19.59 ] [Form_Parse_String] - Exit:
[2011.09.27 10.19.59 ] [Read_configFile()] - Entered:
[2011.09.27 10.20.05 ] [Read_configFile()] - Exit:
[2011.09.27 10.20.05 ] [config_parser.sh] - Exit:
[2011.09.27 10.20.05 ] [AutoOsDetection()] - Exit:
[2011.09.27 10.20.05 ] [packageTest.sh] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Exit:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.25 ] [NFScheck()] - Exit:
[2011.09.27 10.20.25 ] [NFScheck()] - Entered:
[2011.09.27 10.20.26 ] [NFScheck()] - Exit:
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DB2_Version
[2011.09.27 10.20.26 ] Executing: DB2_Version.sh
[2011.09.27 10.20.26 ] Finished: DB2_Version.sh
[2011.09.27 10.20.26 ] Done    : DB2_Version
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DBType
[2011.09.27 10.20.26 ] Executing: DBType
[2011.09.27 10.20.26 ] Finished: DBType
[2011.09.27 10.20.26 ] Done    : DBType
[2011.09.27 10.20.26 ] Starting: DB2_Version
[2011.09.27 10.20.26 ] Executing: DB2_Version.sh
[2011.09.27 10.20.26 ] Finished: DB2_Version.sh
[2011.09.27 10.20.26 ] Done    : DB2_Version
[2011.09.27 10.20.26 ] Starting: DBType
```

Figure 15. fichier prs.trc sur les systèmes UNIX

Problèmes d'exécution

Vous pouvez utiliser la liste de contrôle des problèmes d'exécution pour résoudre les erreurs que vous pouvez rencontrer lors de l'exécution de Prerequisite Scanner.

Exécutez le script Prerequisite Scanner avec les paramètres d'entrée facultatifs **debug** et **trace** pour faciliter le débogage des anomalies.

Tableau 13. Liste de contrôle des problèmes d'exécution

Contrôle	Article
☐	Lorsque vous définissez le paramètre **outputDir** sur la ligne de commande et que le répertoire de sortie n'existe pas, Prerequisite Scanner le crée. Vous devez disposer des droits en écriture pour créer ou écrire dans le répertoire de sortie dans lequel Prerequisite Scanner enregistre les fichiers. Si vous ne disposez pas des droits en écriture, le message d'erreur suivant s'affiche dans l'interface de ligne de commande : `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	Avant d'exécuter Prerequisite Scanner, vérifiez que le disque sur lequel vous souhaitez exécuter Prerequisite Scanner et dans le répertoire de sortie duquel vous souhaitez enregistrer les résultats n'est pas plein ; sinon, le message d'erreur suivant s'affiche dans l'interface de ligne de commande : `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	Si Prerequisite Scanner génère un code de retour de 2, il se peut qu'une erreur de syntaxe de script ou de collecteur se soit produite. Etudiez les causes associées à ce code d'erreur. En cas d'erreur de syntaxe de script, réexécutez Prerequisite Scanner à l'aide de la syntaxe correcte.

Concepts associés :

L'écriture de messages dans les fichiers journaux est désactivée par défaut sur les systèmes UNIX. Vous pouvez activer les fonctions de débogage et de trace à l'aide des paramètres d'entrée **debug** et **trace**. Le scanner écrit les données de débogage et de trace dans différents fichiers journaux et utilise des horodatages pour indiquer les heures de début et de fin des étapes ou des fonctions. Vous pouvez utiliser les deux fichiers pour mettre en corrélation et résoudre une anomalie, une fonction ou un contrôle de prérequis spécifique.

Prerequisite Scanner génère des codes de retour dépendant des résultats de l'analyse et du fait qu'il faille éventuellement quitter le système en raison d'erreurs. Ces codes de retour sont écrits dans les fichiers journaux.

Le script `prereq_checker` exécute IBM Prerequisite Scanner et vérifie les prérequis en fonction du jeu de paramètres que vous spécifiez lorsque vous exécutez le script.

Codes de retour

Prerequisite Scanner génère des codes de retour dépendant des résultats de l'analyse et du fait qu'il faille éventuellement quitter le système en raison d'erreurs. Ces codes de retour sont écrits dans les fichiers journaux.

Prerequisite Scanner génère des codes de retour en se basant sur un ensemble de résultats définis comme suit :

Code de retour	Description
`0`	Renvoie `0` lorsque Prerequisite Scanner est exécuté avec succès et que tous les résultats d'analyse sont `PASS`.
`1`	Renvoie `1` lorsque Prerequisite Scanner est exécuté avec succès, mais qu'un ou plusieurs contrôles des conditions prérequises renvoient `FAIL`.
`2`	Renvoie `2` lorsque Prerequisite Scanner n'est pas exécuté avec succès et doit fermer en raison d'une erreur classée de la manière suivante : • Erreurs de syntaxe de script • Erreurs de collecteur • Autres erreurs

Erreurs de syntaxe de script

Prerequisite Scanner peut se fermer en raison de l'une des erreurs de syntaxe suivantes lors de l'exécution du script :

- Le paramètre d'entrée **`Product_Code`** n'est pas valide ; par exemple il est introuvable ou n'est pas dans un format pris en charge.
- Le modèle de **`Product_Code`** et les paramètres d'entrée **`Product_Version`** ne sont pas valides ; par exemple plusieurs codes et versions sont fournis entre guillemets ou bien le modèle n'est pas noté entre guillemets.
- Les paramètres d'entrée **`Product_Version`** ne sont pas valides ; par exemple la version du produit ne contient pas que des caractères numériques.
- Aucun paramètre d'entrée n'a été saisi dans l'interface de ligne de commande.
- La syntaxe était incorrecte lors de la saisie dans l'interface de ligne de commande ; par exemple un argument de ligne de commande non pris en charge a été saisi.
- Aucun paramètre d'entrée **`Product_Code`** obligatoire n'a été saisi.

Erreurs de collecteur

Prerequisite Scanner peut se fermer en raison de l'une des erreurs de collecteur suivantes :

- Le fichier de résultats temporaire du collecteur est introuvable dans le répertoire *ips_output_dir*`/temp`.
- Le fichier script du collecteur ne s'est pas correctement exécuté.

Autres erreurs

Prerequisite Scanner peut se fermer car l'utilisateur ne dispose pas des droits en écriture nécessaires pour le répertoire de sortie *ips_output_dir*.

Concepts associés :

IBM Prerequisite Scanner génère une sortie pour l'écran et les formats de fichier lisibles suivants : sortie vers l'interface de ligne de commande, fichiers journaux de débogage et de trace, et fichiers texte et XML pour les résultats.

Annexe A. Référence aux codes produit

IBM Prerequisite Scanner utilise un code à plusieurs caractères *product_code* pour identifier le produit, la plateforme individuelle prise en charge et la version du système d'exploitation. Le fichier *ips_root*/codename.cfg contient les paires de valeurs de nom représentant le code du produit, sa plateforme prise en charge et la version du système d'exploitation.

tableau 14 présente l'ensemble actuel de codes produit prédéfinis.

Restriction : IBM Tivoli Monitoring et Tivoli Composite Application Manager possèdent des codes produit prédéfinis que Prerequisite Scanner considère comme réservés. Ces codes ne doivent pas être utilisés comme des codes produit Prerequisite Scanner à moins qu'ils ne fassent référence à leurs agents IBM Tivoli Monitoring et Tivoli Composite Application Manager associés. Pour plus d'informations sur les codes produit, voir ITM 6.X Product Codes Technote.

Restriction : Sur UNIX seulement : Lorsque vous entrez la valeur du code produit dans le fichier, évitez d'utiliser pour. C'est un mot réservé qui peut avoir un impact sur le fonctionnement de Prerequisite Scanner.

Tableau 14. Codes produit prédéfinis

Code produit prédéfini	Plateforme	Version du produit, plateforme, système d'exploitation
ADE	Tous	Autonomic Deployment Engine
BSM	Tous	Tivoli Business Service Manager
CDB	Tous	Tivoli Composite Application Manager (ITCAM) for Applications : DB2
COA	UNIX	Tivoli Provisioning Manager for UNIX
COB	AIX	Tivoli Provisioning Manager for AIX
COC	AIX	Tivoli Provisioning Manager for AIX V5.3.0.0 {64 bits}
COD	AIX	Tivoli Provisioning Manager for AIX 6.1
COE	Linux	Tivoli Provisioning Manager for Linux
COF	Linux	Tivoli Provisioning Manager for Red Hat Linux
COG	Linux	Tivoli Provisioning Manager Version 7.2 for Red Hat Enterprise Linux 5 x86 64 bits
COH	Linux	Tivoli Provisioning Manager for Red Hat Enterprise Linux 5 System z 64 bits
COI	Linux	Tivoli Provisioning Manager for SUSE 10
COJ	Solaris	Tivoli Provisioning Manager Version 7.2 for Solaris
COK	HP-UX	Tivoli Provisioning Manager Version 7.2 for HP-UX
COL	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE zSeries 10
COM	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE 11
CON	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE zSeries 11

Tableau 14. Codes produit prédéfinis (suite)

Code produit prédéfini	Plateforme	Version du produit, plateforme, système d'exploitation
COX	Windows	Tivoli Provisioning Manager Version 7.2 for Windows 2008
COY	Windows	Tivoli Provisioning Manager Version 7.2 for Windows 2003
COZ	Windows	Tivoli Provisioning Manager Version 7.2 for Windows
DMO	Tous	Démonstration Prerequisite Scanner
GYM	UNIX	IBM Tivoli Netcool Performance Manager
KCJ	Windows	Tivoli Enterprise Portal Client
	UNIX	Tivoli Enterprise Portal Client for UNIX
KCQ	Windows	Tivoli Enterprise Portal Server
	UNIX	Tivoli Enterprise Portal Server for UNIX
KHD	Tous	Warehouse Proxy Agent
KHE	UNIX	Warehouse Proxy Agent for UNIX
KIS	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions : Internet Service Monitoring
KLZ	UNIX	Tivoli Monitoring Operating System Agent for Linux
KM6	Windows	IBM Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition
KMQ	Tous	Tivoli Composite Application Manager Agent for WebSphere MQ
KMS	Windows	Tivoli Enterprise Monitoring Server
	UNIX	Tivoli Enterprise Monitoring Server for UNIX
KNT	Windows	Tivoli Monitoring Operating System Agent for Windows
	UNIX	Windows OS monitoring Agent for UNIX
KOR	Windows	Tivoli Monitoring Agent for Oracle
KQI	Tous	Tivoli pourWebSphere Message Broker
KSY	Windows	Summarization and Pruning Agent
	UNIX	Summarization and Pruning Agent for UNIX
KUD	Windows	Tivoli Monitoring Agent for DB2
	UNIX	Tivoli Monitoring Agent for DB2
KTO	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Reporter
KTU	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Collector
KT3	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Application Management Console
KT4	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Client Response Time
KT5	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Web Response Time
KT6	Tous	Tivoli Composite Application Manager (ITCAM) for Transactions : Robotic Response Time

Tableau 14. Codes produit prédéfinis (suite)

Code produit prédéfini	Plateforme	Version du produit, plateforme, système d'exploitation
KZE	Tous	Tivoli zEnterprise Monitoring Agent
LCM	Windows	Tivoli License Compliance Manager
	UNIX	Tivoli License Compliance Manager for UNIX
NCI	Tous	Tivoli Netcool/Impact
NOC	Tous	Composants serveur et composant de bureau Tivoli Netcool/OMNIbus
NOD	Tous	Composant de bureau Tivoli Netcool/OMNIbus
NOS	Tous	Composants serveur Tivoli Netcool/OMNIbus
PAE	Tous	Tivoli Process Automation Engine
TAD	Windows	Tivoli Asset Discovery for Distributed
	UNIX	Tivoli Asset Discovery for Distributed for UNIX
TCR	Tous	Tivoli Common Reporting
TPM	Tous	Tivoli Provisioning Manager

Annexe B. Référence des fichiers de configuration

The IBM Prerequisite Scanner fournit un ensemble prédéfini de fichiers de configuration que vous pouvez éditer. Ces fichiers se trouvent dans *ips_root*/UNIX_Linux ou dans *ips_root*/Windows. Les fichiers ont comme extension .cfg.

tableau 15 répertorie les fichiers de configuration prédéfinis actuellement pris en charge.

Tableau 15. Fichiers de configuration prédéfinis

Fichier de configuration	Plateforme	Version, plateforme, système d'exploitation du produit
ADE_01040000.cfg	Toutes	Autonomic Deployment Engine Version 1.4
BSM_04210000.cfg	Toutes	Tivoli Business Service Manager Version 4.2.1
BSM_06100000.cfg	Toutes	Tivoli Business Service Manager Version 6.1
CDB_06220000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Applications : DB2 Version 6.2.2
COA_07200000.cfg	UNIX	Tivoli Provisioning Manager Version 7.2 for UNIX
COB_07200000.cfg	AIX	Tivoli Provisioning Manager Version 7.2 for AIX
COC_07200000.cfg	AIX	Tivoli Provisioning Manager Version 7.2 for AIX V5.3.0.0 {64 bit}
COD_07200000.cfg	AIX	Tivoli Provisioning Manager Version 7.2 for AIX 6.1
COE_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for Linux
COF_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for Red Hat Linux
COG_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for Red Hat Enterprise Linux 5 x86 64 bits
COH_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for Red Hat Enterprise Linux 5 System z 64 bits
COI_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE 10
COJ_07200000.cfg	Solaris	Tivoli Provisioning Manager Version 7.2 for Solaris
COK_07200000.cfg	HP-UX	Tivoli Provisioning Manager Version 7.2 for HP-UX
COL_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE zSeries 10
COM_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE 11
CON_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE zSeries 11
COX_07200000.cfg	Windows	Tivoli Provisioning Manager Version 7.2 for Windows 2008
COY_07200000.cfg	Windows	Tivoli Provisioning Manager Version 7.2 for Windows 2003
COZ_07200000.cfg	Windows	Tivoli Provisioning Manager Version 7.2 for Windows
DMO_00000000.cfg	Toutes	Prerequisite Scanner demo
DMO_01000000.cfg	Toutes	Prerequisite Scanner Version 1.0 demo
GYM_01030200.cfg	UNIX	IBM Tivoli Netcool Performance Manager Version 1.3.2
KCJ_06200000.cfg	Windows	Tivoli Enterprise Portal Client Version 6.2
KCJ_06210000.cfg	UNIX	Tivoli Enterprise Portal Client Version 6.2.1
KCJ_06220000.cfg	Toutes	Tivoli Enterprise Portal Client Version 6.2.2

Tableau 15. Fichiers de configuration prédéfinis (suite)

Fichier de configuration	Plateforme	Version, plateforme, système d'exploitation du produit
KCQ_06200000.cfg	Windows	Tivoli Enterprise Portal Server Version 6.2
KCQ_06210000.cfg	UNIX	Tivoli Enterprise Portal Server Version 6.2.2
KCQ_06220000.cfg	Toutes	Tivoli Enterprise Portal Server Version 6.2.2
KHD_06200000.cfg	Windows	Warehouse Proxy Agent Version 6.2
KHD_06210000.cfg	Toutes	Warehouse Proxy Agent Version 6.2.1
KHD_06220000.cfg	Toutes	Warehouse Proxy Agent Version 6.2.2
KHE_06220000.cfg	UNIX	Warehouse Proxy Agent Version 6.2.2
KIS_07200000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Internet Service Monitoring Version 7.2
KIS_07300000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Internet Service Monitoring Version 7.3
KLZ_06210000.cfg	UNIX	Tivoli Monitoring Operating System Agent for Linux Version 6.2.1
KLZ_06220000.cfg	UNIX	Tivoli Monitoring Operating System Agent for Linux Version 6.2.2
KM6_070100000.cfg	Windows	Tivoli Composite Application Manager Agent for WebSphere MQ File Transfer Edition Version 7.1
KMQ_070100000.cfg	Toutes	Tivoli Composite Application Manager Agent for WebSphere MQ Version 7.1
KMS_06200000.cfg	Windows	Tivoli Enterprise Monitoring Server Version 6.2
KMS_06210000.cfg	Toutes	Tivoli Enterprise Monitoring Server Version 6.2.1
KMS_06220000.cfg	Toutes	Tivoli Enterprise Monitoring Server Version 6.2.2
KNT_06200000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows Version 6.2
KNT_06210000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows Version 6.2.1
KNT_06220000.cfg	Windows	Tivoli Monitoring Operating System Agent for Windows Version 6.2.2
KOR_06220000.cfg	Windows	Tivoli Monitoring Agent for Oracle Version 6.2.2
KQI_07010000.cfg	Toutes	Tivoli pourWebSphere Message Broker Version 7.1
KSY_06200000.cfg	Windows	Summarization and Pruning Agent Version 6.2
KSY_06210000.cfg	Toutes	Summarization and Pruning Agent Version 6.2.1
KSY_06220000.cfg	Toutes	Summarization and Pruning Agent Version 6.2.2
KTO_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Reporter Version 7.2
KTO_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Reporter Version 7.2.2
KTO_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Reporter Version 7.3
KTU_07200000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Collector Version 7.2
KTU_07200200.cfg	Windows	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Collector Version 7.2.2
KTU_07300000.cfg	UNIX	Tivoli Composite Application Manager (ITCAM) for Transactions : Transaction Collector Version 7.3
KT3_07300000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Application Management Console Version 7.3

Tableau 15. Fichiers de configuration prédéfinis (suite)

Fichier de configuration	Plateforme	Version, plateforme, système d'exploitation du produit
KT4_07300000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Client Response Time Version 7.3
KT5_07300000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Web Response Time Version 7.3
KT6_07300000.cfg	Toutes	Tivoli Composite Application Manager (ITCAM) for Transactions : Robotic Response Time Version 7.3
KUD_06100000.cfg	Windows	Tivoli Monitoring Agent for DB2 Version 6.1
KUD_06200000.cfg	Toutes	Tivoli Monitoring Agent for DB2 Version 6.2
KUD_06210000.cfg	Toutes	Tivoli Monitoring Agent for DB2 Version 6.2.1
KUD_06220000.cfg	Toutes	Tivoli Monitoring Agent for DB2 Version 6.2.2
KZE_06020300.cfg	Toutes	Tivoli zEnterprise Monitoring Agent Version 6.2.3
LCM_01000000.cfg	Toutes	Tivoli License Compliance Manager Version 1.0
LCM_02300000.cfg	Toutes	Tivoli License Compliance Manager Version 2.3
NCI_06100000.cfg	Toutes	Tivoli Netcool/Impact Version 6.1
NOC_07310000.cfg	Toutes	Composant serveur et composant de bureau Tivoli Netcool/OMNIbus Version 7.3.1
NOD_07310000.cfg	Toutes	Composant de bureau Tivoli Netcool/OMNIbus Version 7.3.1
NOS_07310000.cfg	Toutes	Composant serveur Tivoli Netcool/OMNIbus Version 7.3.1
PAE_07500000.cfg	Toutes	Tivoli Process Automation Engine
TAD_07200000.cfg	Toutes	Tivoli Asset Discovery for Distributed Version 7.2
TAD_07220000.cfg	Toutes	Tivoli Asset Discovery for Distributed Version 7.2.2
TCR_02010100.cfg	Toutes	Tivoli Common Reporting
TPM_07210000.cfg	Toutes	Tivoli Provisioning Manager Version 7.2.1

Annexe C. Référence de propriétés de prérequis

Cette référence présente les propriétés de prérequis de base pour chaque catégorie prédéfinie des prérequis matériels et logiciels.

tableau 16 présente les catégories prédéfinies des prérequis matériels et logiciels.

Tableau 16. Catégories prédéfinies des propriétés de prérequis

Catégorie de données	Description	Identificateur à préfixe obligatoire	Référence
Eléments communs	Les propriétés de données communes vérifient les prérequis communs tels que la vitesse du processeur, la mémoire vive, le disque et l'espace temporaire.	Aucun	«Propriétés de données communes», à la page 92
Autonomic Deployment Engine	Les propriétés de données Autonomic Deployment Engine vérifient les prérequis Autonomic Deployment Engine tels que l'unité d'installation.	`de`	«Propriétés de données Autonomic Deployment Engine», à la page 97
Logiciel installé	Les propriété de données du logiciel installé vérifient les prérequis du logiciel installé tels que les programmes enregistrés dans le registre Windows et cygwin et gskit sont installés.	Aucun	«Propriétés de données du logiciel installé», à la page 113
Utilisateur	Les propriétés de données utilisateur vérifient les prérequis de l'utilisateur, par exemple que l'utilisateur connecté dispose des droits d'administration ou est le superutilisateur.	`user`	«Propriétés de données utilisateur», à la page 113
Système d'exploitation	Les propriétés de données du système d'exploitation contrôlent les prérequis du système d'exploitation tels que la version, l'architecture, la mémoire totale, la mémoire disponible et la mémoire physique totale.	`os`	«Propriétés de données du système d'exploitation», à la page 101
Connectivité	Les propriétés de données de connectivité vérifient les prérequis de connectivité, par exemple que Telnet est en cours d'exécution ainsi que les adresses IP et ports auxquels le scanner peut être connecté.	Aucun	«Propriétés de données de connectivité», à la page 98
Réseau	Les propriétés de données de réseau vérifient les prérequis de réseau qui peuvent être communs à toutes les plateformes, par exemple s'il y a des ports disponibles.	`network`	«Propriétés de données de réseau», à la page 99
Réseau Windows	Les propriétés de données du réseau Windows vérifient les prérequis du réseau, par exemple que NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING.	`network`	«Propriétés de données du réseau Windows», à la page 114

Tableau 16. Catégories prédéfinies des propriétés de prérequis (suite)

Catégorie de données	Description	Identificateur à préfixe obligatoire	Référence
Réseau UNIX	Les propriétés de données du réseau UNIX vérifient les prérequis du réseau, par exemple que NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING.	`network`	«Propriétés de données du réseau UNIX», à la page 114
Internet Explorer	Les propriétés de données Microsoft Internet Explorer vérifient les prérequises Internet Explorer tels que la version.	`internetExplorer`	«Propriétés de données Internet Explorer», à la page 99
Serveur de base de données, DB2	Les propriétés de données DB2 vérifient les prérequis DB2 tels que la version.	`DB2`	«Propriétés de données DB2», à la page 98
Serveur de base de données, MS SQL	Les propriétés de données du serveur MS SQL vérifient les prérequis du serveur MS SQL tels que la version.	`mssql`	«Propriétés de données du serveur SQL MS», à la page 98
Serveur de base de données, Oracle	Les propriétés de données Oracle vérifient les prérequis Oracle tels que la version.	`Oracle`	«Propriétés de données Oracle», à la page 100
Variables d'environ-nement	Les variables d'environnement vérifient les prérequis des variables d'environnement, par exemple si la variable d'environnement a été définie.	`env`	«Propriétés de données de la variable d'environnement», à la page 115

Propriétés de données communes

Les propriétés de données communes vérifient les prérequis communs tels que la vitesse de l'unité centrale, la mémoire vive, le disque et l'espace temporaire. Pour les systèmes Windows, le système utilise le script principal IBM Prerequisite Scanner. Pour les systèmes UNIX, le système utilise le script principal Prerequisite Scanner et le collecteur commun *ips_root*`/Unix_Linux/common.sh`.

tableau 17 présente les propriétés de prérequis de données communes. Cette catégorie de propriétés de prérequis n'exige pas d'identificateur à préfixe.

Tableau 17. Propriétés de prérequis de données communes

Propriété de prérequis	Plateformes	Description	Valeurs valides
`CPU Name`	Tous	Le nom de l'unité centrale est utilisé uniquement pour l'affichage dans les résultats	Sans objet
`CpuArchitecture`	UNIX	Architecture du système d'exploitation	Chaîne, avec plusieurs valeurs prises en charge séparées par une virgule, par exemple : `x86_64,s390x,ppc64,AMD64`

Tableau 17. Propriétés de prérequis de données communes (suite)

Propriété de prérequis	Plateformes	Description	Valeurs valides

`DBType`	Tous	Vérifie les types de serveurs de base de données installés sur la machine.
		
		Pour Oracle sous les systèmes UNIX uniquement : le collecteur attend que les variables d'environnement `ORACLE_BASE` et `ORACLE_HOME` soient définies dans le fichier `$HOME/.profile`, par exemple :
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		où `$HOME` doit être `/home/oracle`, le répertoire initial de l'utilisateur Oracle.
			Cette valeur peut avoir l'un des types suivants :
			• Chaîne représentant tout type de serveur de base de données, par exemple :
			`any`
			• Chaîne représentant le type de serveur de base de données, par exemple :
			`Oracle`
			• `regex{`*`str`*`}`, expression régulière avec le paramètre d'entrée *str* représentant le modèle de recherche pour le type de serveur de base de données, par exemple :
			`regex{.*MSSQL.*|DB2.*}`
			
			Vérifie que le type de serveur de base de données est SQL MS ou DB2 sur les systèmes Windows.
			• Chaîne ne représentant aucun type de serveur de base de données, par exemple :
			`unknown`

`DBTypeDetails`	Tous	Types de serveurs de base de données installés sur la machine.
		
		Pour Oracle sous les systèmes UNIX uniquement : le collecteur attend que les variables d'environnement `ORACLE_BASE` et `ORACLE_HOME` soient définies dans le fichier `$HOME/.profile`, par exemple :
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		où `$HOME` doit être `/home/oracle`, le répertoire initial de l'utilisateur Oracle.
		
		La propriété de prérequis écrit les informations détaillées concernant le type de serveur de base de données, c'est-à-dire le type de serveur de base de données, l'emplacement d'installation et la version dans le fichier `result.txt`. Les informations détaillées concernant plusieurs types de serveurs de base de données sont séparées par des point-virgules
			Cette valeur peut avoir l'un des types suivants :
			• Chaîne représentant tout type de serveur de base de données, par exemple :
			`any`
			• Chaîne représentant un seul type de serveur de base de données, par exemple :
			`DB2`
			• `regex{`*`str`*`}`, expression régulière avec le paramètre d'entrée *str* représentant le motif recherché pour le type de serveur de base de données, par exemple :
			`regex{.*MSSQL.*|DB2.*}`
			
			Vérifie que le type de serveur de base de données est SQL MS ou DB2 sur les systèmes Windows.

Tableau 17. Propriétés de prérequis de données communes (suite)

Propriété de prérequis	Plateformes	Description	Valeurs valides

`Disk`	Windows	Volume d'espace disque libre avec les attributs de qualification facultatifs suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au répertoire à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
			Cette valeur peut avoir l'un des types suivants :
			• Chaîne présentant le format de qualificatif suivant :
			`[dir:`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			Par exemple :
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			• Format numérique en Mo ou Go :
			`disk_space``MB|GB`
			Par exemple :
			`Disk=250MB`

`Disk`	UNIX	Volume d'espace disque libre	Format numérique en Go ou Mo, par exemple : `2GB`
`intel.cpu`	Tous	Vitesse de l'unité centrale du processeur Intel	Format numérique en GHz et sous Windows uniquement en MHz également, par exemple : `2GHz`
`Memory`	Tous	Taille totale de mémoire physique actuellement disponible sur la machine. **Conseil :** Contrôlez séparément la taille de mémoire physique et virtuelle disponible à l'aide des propriétés de prérequis prédéfinies dans la catégorie du système d'exploitation.	Format numérique en Go ou Mo, par exemple : `300MB`

Tableau 17. Propriétés de prérequis de données communes (suite)

Propriété de prérequis	Plateformes	Description	Valeurs valides

`OS Version`	Tous	Nom complet et version du système d'exploitation en cours d'exécution sur la machine ; vous pouvez également utiliser une expression régulière pour transmettre une chaîne représentant les multiples variantes d'un système d'exploitation.
		Conseil : Utilisez cette propriété de prérequis avec `os.servicePack` et `os.architecture` afin de contrôler le module de mise à jour et l'architecture système actuels.
			Cette valeur peut avoir l'un des types suivants :
			• Chaîne pouvant représenter plusieurs versions, chaque version étant séparée par une virgule, par exemple :
			`RedHat Enterprise Linux 6.*, SuSE Linux Enterprise Server 11, SuSE Linux Enterprise Server 10, SuSE Linux Enterprise Server 9, AIX V6.1,AIX V5.3`
			Restriction : Sur les systèmes Windows, le caractère générique `*` est uniquement pris en charge dans une expression régulière.
			• `regex{str}`, expression régulière avec le paramètre d'entrée *str* représentant le motif recherché pour la version, par exemple :
			`regex{Windows 200[3-8]}`
			Vérifie que le système d'exploitation réel correspond à une version allant de Windows 2003 à Windows 2008.
			`regex{Red Hat*.*}`
			Vérifie que le système d'exploitation réel correspond à une variante de Red Hat Linux.
			Remarque : Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

`numCPU`	Tous	Nombre de coeurs ou de processeurs indépendants sur l'ordinateur. Si l'outil scanne l'ordinateur et ne trouve aucun coeur ou trouve des processeurs qui ne sont pas des coeurs, il renvoie un résultat introuvable.	Nombre, par exemple `4`
`risc.cpu`	UNIX	Vitesse de l'unité centrale d'un processeur RISC	Format numérique en GHz, par exemple : `1.4GHz`
`Temp`	UNIX	Espace disque disponible pour le système de fichiers *Temp* spécifié	Format numérique en Go/s ou Mo/s, par exemple : `300MB`

Concepts associés :

Prerequisite Scanner gère le contrôle de la propriété de condition prérequise `Memory` différemment suivant qu'un agent Tivoli Monitoring ou Tivoli Composite Application Manager est déjà exécuté ou non sur l'ordinateur.

Référence associée :

Les propriétés de données du système d'exploitation contrôlent les prérequis du système d'exploitation tels que la version, l'architecture, la mémoire totale, la mémoire disponible et la mémoire physique totale. Pour les systèmes Windows uniquement, le système utilise les collecteurs VBScript du système d'exploitation dans le répertoire *ips_root*`/lib` avec l'identificateur à préfixe`os` figurant dans leur nom de fichier. Pour les systèmes UNIX uniquement, le système utilise les collecteurs du système d'exploitation UNIX dans le répertoire *ips_root*`/UNIX_Linux` avec l'identificateur à préfixe `os` figurant dans leur nom de fichier.

Comportement du système pour la propriété de conditions prérequise de mémoire et les agents Tivoli Monitoring

Prerequisite Scanner gère le contrôle de la propriété de condition prérequise `Memory` différemment suivant qu'un agent Tivoli Monitoring ou Tivoli Composite Application Manager est déjà exécuté ou non sur l'ordinateur.

Si un agent est déjà installé, Prerequisite Scanner utilise une valeur attendue pour la propriété de prérequis `Memory` en se basant sur la différence entre la valeur attendue des fichiers de configuration nouveaux et existants si le fichier de configuration existant se trouve toujours sur l'ordinateur ; sinon, il gère la valeur attendue par le comportement par défaut.

Lorsque vous exécutez Prerequisite Scanner pour contrôler les prérequis pour un agent Tivoli Monitoring en cours de mise à niveau ou de réinstallation, il contrôle tout d'abord que l'agent est déjà exécuté sur l'ordinateur. Si l'agent est en cours d'exécution, Prerequisite Scanner recherche le fichier de configuration associé à la version existante de l'agent en cours d'exécution. Le comportement suivant se produit suivant le résultat de cette recherche :

- Si le système ne parvient pas à trouver le fichier de configuration, Prerequisite Scanner part du principe que l'environnement cible n'a pas été analysé auparavant ; c'est pourquoi Prerequisite Scanner utilise la valeur attendue pour la propriété de prérequis `Memory` spécifiée dans le nouveau fichier de configuration qui suit le comportement par défaut. Prerequisite Scanner écrit cette valeur attendue dans le résultat de sortie.
- Si le système trouve le fichier de configuration, Prerequisite Scanner compare la valeur attendue de la propriété de prérequis `Memory` de la version existante avec la valeur attendue figurant dans le fichier de configuration de la nouvelle version. S'il y a une différence entre les valeurs, et que la nouvelle valeur est supérieure à la valeur attendue existante, Prerequisite Scanner définit cette différence comme la valeur attendue. Prerequisite Scanner écrit le différentiel de valeurs attendues dans le résultat de sortie. Par exemple, le fichier de configuration de la version 1 de l'agent indique 1 Go comme valeur attendue. Le nouveau fichier de configuration de la version 2 de l'agent indique 1,5 Go comme valeur attendue ; c'est pourquoi Prerequisite Scanner utilise et écrit 0,5 Go comme différentiel de valeurs attendues.

Propriétés de données Autonomic Deployment Engine

Les propriétés de données Autonomic Deployment Engine vérifient les prérequis Autonomic Deployment Engine tels que l'unité d'installation. Pour les systèmes Windows uniquement, le système utilise les collecteurs Autonomic Deployment Engine dans le répertoire *ips_root*/lib/, avec le préfixe de figurant dans leur nom de fichier. Pour les systèmes UNIX uniquement, le système utilise les collecteurs UNIX Autonomic Deployment Engine dans le répertoire *ips_root*/UNIX_Linux avec le préfixe de figurant dans leur nom de fichier.

tableau 18 présente les propriétés de prérequis. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe de.

Tableau 18. Propriétés de données Autonomic Deployment Engine

Propriété de prérequis	Plateforme	Description	Valeurs valides
de.installed	Tous	Vérifie que le produit est installé.	Valeur booléenne, par exemple : true|false
de.installationUnit	Tous	Vérifie que l'unité d'installation définie est installée à l'aide de la commande **listIU** - v	Cette value peut avoir l'un des types suivants : • Chaîne représentant une unité d'installation unique, par exemple l'unité d'installation de Tivoli Integrated Portal : C37109911C8A11D98E1700061BDE7AEA, B24209911C8A11D98E1700061BDE7AEA • Chaîne permettant de représenter plusieurs unités d'installation, par exemple : 5FFE79F918DF3BA0D67511FD3F7C358E • regex {str}, expression régulière avec le paramètre d'entrée str représentant le modèle de recherche pour l'unité d'installation, la version et le chemin d'installation ; par exemple pour contrôler l'unité d'installation, la version de WebSphere Application Server et le chemin d'installation de Tivoli Integrated Portal, le modèle de recherche se présente comme suit : • regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *C:\\IBM\\tivoli\\tip.*} **Remarque :** Vous pouvez également utiliser une variable d'environnement pour le chemin d'installation ; par exemple en remplaçant le chemin par la variable d'environnement TIPHOME, le motif recherché est le suivant : regex{.*C00DA95AFD9B7E0397153CD944B5A255. *6.1.0.2100.*SIU eWAS. *%TIPHOME%.*} • Plusieurs arguments regex {str} permettant de représenter plusieurs contrôles ; par exemple : regex{.*C37109911C8A11D98E1700061BDE7AEA.*}, regex{.*B24209911C8A11D98E1700061BDE7AEA.*}

Propriétés de données de connectivité

Les propriétés de données de connectivité vérifient les prérequis de connectivité, par exemple que Telnet est en cours d'exécution et à quelles adresses IP et quels ports le scanner peut être connecté. Pour les systèmes Windows uniquement, le système utilise le collecteur de connectivité *ips_root*/lib/connectivity_plug.vbs. Pour les systèmes UNIX, le système utilise le script principal IBM Prerequisite Scanner et le collecteur de connectivité *prs_root*/Unix_Linux/connectivity_plug.sh. La sortie est transmise uniquement au fichier journal de débogage.

Propriétés de données DB2

Les propriétés de données DB2 vérifient les prérequis DB2 tels que la version. Pour les systèmes Windows uniquement, le système utilise le collecteur DB2 dans le répertoire *ips_root*/lib/db2_version_plug.bat. Pour les systèmes UNIX uniquement, le système utilise les collecteurs UNIX DB2 dans le répertoire *ips_root*/UNIX_Linux avec le préfixe db2 figurant dans leur nom de fichier.

tableau 19 présente les propriétés de prérequis DB2. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe DB2.

Tableau 19. Propriétés de données DB2

Propriété de prérequis	Plateforme	Description	Valeurs valides
DB2 Version	Tous	Version de DB2 actuellement installée sur la machine	Chaîne, par exemple : v9.5.100.179FP4
db2.home.space	UNIX	Espace disque disponible pour le répertoire initial DB2	Format numérique en Go, par exemple : 8 Go

Propriétés de données du serveur SQL MS

Les propriétés de données du serveur SQL MS vérifient les prérequis du serveur SQL MS tels que la version et l'emplacement. Pour les systèmes Windows uniquement, le système utilise les collecteurs du serveur MS SQL dans le répertoire *ips_root*/Windows/, avec le préfixe mssql figurant dans leur nom de fichier.

tableau 20 présente les propriétés de prérequis du serveur SQL MS. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe mssql.

Tableau 20. Propriétés de données du serveur SQL MS

Propriété de prérequis	Plateforme	Description	Valeurs valides
mssql.Client	Windows	Vérifie la version du client SQL MS actuellement installée sur la machine	La valeur de chaîne attendue peut correspondre à plusieurs versions, séparées par une virgule, par exemple : 10.50.1600.1 **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

Tableau 20. Propriétés de données du serveur SQL MS (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`mssql.Server`	Windows	Vérifie la version du serveur SQL MS actuellement installée sur la machine	La valeur de chaîne attendue peut correspondre à plusieurs versions, séparées par une virgule, par exemple : `10.50.1600.1` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`mssql.Server.Location`	Windows	Vérifie le répertoire initial de la base de données SQL MS	Chaîne, par exemple : `any`

Propriétés de données Internet Explorer

Les propriétés de données Microsoft Internet Explorer vérifient les prérequis Internet Explorer tels que la version. Le système utilise le collecteur Internet Explorer *ips_root*`/lib/internetExplorer_plug.vbs`.

tableau 21 présente les propriétés de prérequis Internet Explorer. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `internetExplorer`.

Tableau 21. Propriétés de données Internet Explorer

Propriété de prérequis	Description	Valeurs valides
`internetExplorer.version`	Version d'Internet Explorer installée sur la machine	Format numérique, par exemple `7.0+` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

Propriétés de données de réseau

Les propriétés de données de réseau vérifient les prérequis de réseau qui peuvent être communs à toutes les plateformes, par exemple s'il y a des ports disponibles. Le système utilise les collecteurs de réseau dans le répertoire *ips_root*`/lib` avec l'identificateur à préfixe `network` dans leur nom de fichier.

tableau 22, à la page 100 présente les propriétés de prérequis de réseau communs à toutes les plateformes. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `network`.

Tableau 22. Propriétés de données de réseau

Propriété de prérequis	Plateforme	Description	Valeurs valides
`network.availablePorts.` *`app_type`*	Tous	Utilisez cette convention d'attribution de nom pour vérifier que le port ou la gamme de ports est disponible pour le type d'application *app_type*. Vérifiez quels ports ne sont pas en mode écoute, par exemple : • `network.availablePorts.DB2` contrôle les ports du serveur de base de données DB2, *app_type* correspondant à `DB2` • `network.availablePorts.WAS` contrôle les ports pour WebSphere Application Server, où *app_type* est `WAS`	Entiers positifs, par exemple : • `network.availablePorts.DB2 = 50000-50005` • `network.availablePorts.WAS = 8080` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`network.portsInUse.` *`app_type`*	Tous	Utilisez cette convention d'attribution de nom pour vérifier que le port ou la gamme de ports est utilisée pour le type d'application *app_type*. Vérifiez quels ports sont en mode écoute, par exemple : • `network.availablePorts.DB2` contrôle les ports du serveur de base de données DB2, où *app_type* est `DB2` • `network.availablePorts.WAS` contrôle les ports pour WebSphere Application Server, où *app_type* est `WAS`	Entiers positifs, par exemple : • `network.portsInUse.DB2 = 50900-50905` • `network.portsInUse.WAS = 8080` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`network.validateHostsFile`	Windows	Vérifie que toutes les machines hôtes répertoriées dans le fichier hosts sont au format : *`IP_Address Host_Name Short_Name`* où : • *`IP_Address`* correspond au protocole IP de l'ordinateur, par exemple `127.0.0.1` • *`Host_Name`* correspond au nom de système hôte qualifié complet de l'ordinateur, par exemple `localhost.localdomain` • *`Short_Name`* est le nom abrégé de l'ordinateur, par exemple `localhost`	Valeur booléenne, par exemple `True`

Propriétés de données Oracle

Les propriétés de données Oracle vérifient les prérequis Oracle tels que la version. Pour les systèmes Windows uniquement, le système utilise le collecteur Oracle. Pour les systèmes UNIX uniquement, le système utilise les collecteurs Oracle UNIX dans le répertoire *`ips_root`*`/UNIX_Linux` avec le préfixe `oracle` figurant dans leur

nom de fichier. Pour les systèmes Windows uniquement, le système utilise les collecteurs Oracle Windows dans le répertoire *ips_root*/lib avec le préfixe oracle figurant dans leur nom de fichier.

tableau 23 présente les propriétés de prérequis Oracle. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe oracle.

Tableau 23. Propriétés de données Oracle

Propriété de prérequis	Plateforme	Description	Valeurs valides
ORACLE Version	Windows	Vérifie la version d'Oracle actuellement installée sur la machine	La valeur de chaîne attendue peut correspondre à plusieurs versions, séparées par une virgule, par exemple : 9.2, 10.1, 10.2
oracle.Client	Tous	Vérifie la version du client Oracle actuellement installée sur la machine	La valeur de chaîne attendue peut correspondre à plusieurs versions, séparées par une virgule, par exemple : 9.2.0.8+ **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
oracle.Client.Location	Tous	Vérifie le répertoire initial du client Oracle	Chaîne, par exemple : /opt/oracle/products/10.1.0/client_1
oracle.Server	Tous	Vérifie la version du serveur Oracle actuellement installée sur la machine	La valeur de chaîne attendue peut correspondre à plusieurs versions, séparées par une virgule, par exemple : 10.2.0.4g,11g R1 **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
oracle.Server.Location	Tous	Vérifie le répertoire initial du serveur de base de données Oracle	Chaîne, par exemple : /opt/oracle/product/10.1.0/Db_1

Propriétés de données du système d'exploitation

Les propriétés de données du système d'exploitation contrôlent les prérequis du système d'exploitation tels que la version, l'architecture, la mémoire totale, la mémoire disponible et la mémoire physique totale. Pour les systèmes Windows uniquement, le système utilise les collecteurs VBScript du système d'exploitation dans le répertoire *ips_root*/lib avec l'identificateur à préfixeos figurant dans leur nom de fichier. Pour les systèmes UNIX uniquement, le système utilise les collecteurs du système d'exploitation UNIX dans le répertoire *ips_root*/UNIX_Linux avec l'identificateur à préfixe os figurant dans leur nom de fichier.

tableau 24, à la page 102 présente les propriétés de prérequis du système d'exploitation. Cette catégorie de propriétés de prérequis nécessite l'identificateur à préfixe os.

Tableau 24. Propriétés de données du système d'exploitation

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.architecture`	Tous	Vérifie l'architecture du système	`32-bit|64-bit`
`os.automount`	UNIX	Vérifie que les fonctions de montage automatique fonctionnent	Valeur booléenne, par exemple : `True`
`os.autoUpdateEnabled`	Windows	Vérifie que la fonction Windows Update est automatiquement activée ; renvoie `True` le cas échéant	Valeur booléenne, par exemple : `True`
`os.availableMemory`	Windows	Vérifie la taille de la mémoire virtuelle actuellement disponible, mais inutilisée par le système d'exploitation	Format numérique en Mo, par exemple : `900 Mo`

`os.dir.`*`dir_name`*	UNIX	Vérifie le système de fichiers *dir_name* en fonction des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le système de fichiers à contrôler
		• attribut `type` permettant de déterminer l'attribut de système de fichiers à contrôler, par exemple la représentation numérique octale *`octal_digits`* pour les autorisations d'accès à ce système de fichiers
		dir_name peut représenter par exemple :
		• `tmp`
		• `home`
			Chaîne présentant le format de qualificatif suivant :
			`[dir:`*`dir_name`*`, type:permission] `*`octal_digits`*`+`
			Par exemple, pour vérifier que le répertoire initial dispose des autorisations `drwxr-xr-x` :
			`os.dir.home=[dir:/home, type:permission]755+`

`os.diskquota`		Vérifie le quota d'utilisation de disque pour l'utilisateur connecté ; renvoie la valeur du quota en kilooctets ou `Unlimited`	Cette valeur peut avoir l'un des types suivants : • Nombre représentant les kilooctets, par exemple `414000` • Chaîne représentant un quota de disque illimité, par exemple `Unlimited`
`os.expectLink`	UNIX	Vérifie que l'extension Expect pour TCL est disponible sur la machine ; retourne `Available` si la machine présente un statut disponible **Remarque :** La propriété de prérequis `os.file.expect` vérifie que l'extension Expect est installée sur la machine.	`Available|Unavailable`
`os.file.`*`script_name`*	UNIX	Vérifie que le script *script_name* est disponible sur la machine. *script_name* peut représenter par exemple : • `bash` • `expect` • `gzip` • `tar`	Valeur booléenne, par exemple : `True`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.Firefox`	UNIX	Vérifie que Mozilla Firefox est installé sur la machine ; renvoie `Available` si c'est le cas	`Available|Unavailable`
`os.FreePagingSpace`	UNIX	Vérifie la taille totale de la mémoire cache disponible de la page	Format numérique en Mo ou Go, par exemple : `4 Go+` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`os.ftpusers`	UNIX	Vérifie que le superutilisateur est répertorié dans le fichier `ftpusers` qui indique les utilisateurs auxquels les privilèges de connexion ftp ne sont pas accordés ; renvoie `Available` si l'utilisateur n'est pas répertorié	`Available|Unavailable`
`os.gnu.tar`	UNIX	Vérifie que l'utilitaire d'archivage sur bande GNU est disponible sur la machine ; renvoie `Available` s'il est installé	`Available|Unavailable`
`os.hostformat`	UNIX	Vérifie que les entrées de `/etc/host` se présentent au format correct	Valeur booléenne, par exemple : `True`
`os.iodevicestatus`	AIX	Vérifie le statut de l'E-S asynchrone (`aio0`), c'est-à-dire le processus du noyau permettant d'améliorer les performances de l'opération d'E-S ; renvoie `Available` si le processus présente un statut disponible	`Available|Unavailable`
`os.is8dot3File FormatEnabled`	Windows	Vérifie que les formats de nom de fichier 8.3 sont automatiquement appliqués ; renvoie `True` si c'est le cas	Valeur booléenne, par exemple : `True`
`os.localhost InHostsFile`	Tous	Vérifie si une entrée du fichier hosts permet de mapper le système hôte local à l'adresse IP `127.0.0.1`, par exemple : `127.0.0.1 localhost`	Valeur booléenne, par exemple : `True`
`os.isService Running.`*`service_name`*	Windows	Utilisez cette convention d'attribution de nom pour vérifier que *service_name* est en cours d'exécution sur la machine. *nom_service* peut représenter par exemple : • `remoteRegistry` pour Remote Registry Service • `DNSClient` pour services client DNS • `terminalServices` pour Remote Desktop Services ou Terminal Services	Valeur booléenne, par exemple : `True`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.kernelMode`	AIX	Vérifie l'architecture de l'unité centrale prenant en charge le noyau ou le mode illimité	`32-bit|64-bit`
`os.kernelParameters`	Linux	Vérifie que les paramètres de noyau sont disponibles pour le système d'exploitation	`Available|Unavailable`
`os.kernelversion`	Linux	Vérifie l'édition du noyau pour les systèmes d'exploitation Linux	Format numérique, par exemple `2.6`
`os.largeFile`	UNIX	Vérifie la prise en charge des fichiers de grande taille	Valeur booléenne, par exemple : `True`
`os.ldLibPath`	UNIX	Vérifie que la variable d'environnement `LD_LIBRARY_PATH` existe et se termine par deux points, à savoir `os.ldLibPath=[endsWith=:]`	`Available|Unavailable`
`os.level`	AIX	Vérifie que le système d'exploitation AIX est d'un niveau supérieur à 10 pour AIX version 5.3 ou ultérieure et d'un niveau supérieur à 3 pour AIX version 6.1	Valeur booléenne, par exemple : `True`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.lib.`*`lib_name_version`*	UNIX	Vérifie que la version prise en charge de la bibliothèque *lib_name* est installée sur la machine. Chaîne ou expression régulière permettant de représenter *lib_name_version*, par exemple, en gras : • Bibliothèque 32 bits **`libstdc++.so.#`** • Bibliothèque 64 bits **`libstdc++.so.#`** • Bibliothèque 32 bits **`libXft.so.#`** • Bibliothèque 32 bits **`libXtst.so.#`** • bibliothèque 64 bits **`libaio.so.#`** • niveau d'exécution XLC **`xlC.rte`** 32 bits • 32-bit **`xlC.aix50.rte`** exécution XLC pour la version 5.3 AIX • 32-bit **`xlC.aix61.rte`** exécution XLC pour la version 6.1 AIX • Bibliothèque AIX IOCP **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, ensemble de fichiers de code ISO pour le système d'exploitation de base AIX	Cette valeur peut avoir l'un des types suivants : Chaîne, par exemple : • `/usr/lib/libstdc++.so.#` en tant que valeur pour la bibliothèque `libstdc++.so.#` 32 bits • `/usr/lib64/libaio.so.#` en tant que valeur pour la bibliothèque `libaio.so.#` 64 bits • `xlC.aix50.rte.9.0.0.8+` en tant que valeur pour l'exécution XLC `xlC.aix50.rte` 32 bits pour AIX version 5.3 • `bos.loc.iso.en_us` pour l'ensemble de fichiers de code ISO `regex {`*`str`*`}`, expression régulière avec le paramètre d'entrée *str* représentant le motif recherché pour le nom de la bibliothèque, par exemple : `regex{.*libgcc.*}` Vérifie si une version de la bibliothèque d'exécution de bas niveau GCC `libgcc` pour ce système d'exploitation existe. **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`os.loginVariable`	UNIX	Vérifie que les chemins par défaut pour le superutilisateur sont définis dans les variables `PATH` et `SUPATH` ; renvoie `Available` si c'est le cas	`Available|Unavailable`
`os.maximoDirectory`	UNIX	Vérifie que le répertoire `/export/home/maximo` est disponible	`Available|Unavailable`
`os.maximoDirOwner`	UNIX	Vérifie le propriétaire du répertoire `/export/home/maximo`	`maximo`
`os.maximumProcesses`	UNIX	Vérifie le nombre maximum de processus pouvant être exécutés pour chaque utilisateur	Nombre, par exemple `2048`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.MozillaVersion`	UNIX	Vérifie une version spécifique de Mozilla Firefox sur la machine, contrairement à la propriété de prérequis `os.Firefox`	Format numérique, par exemple `3.0+` **Remarque :** . Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

`os.mountcheck`	UNIX	Vérifie que le système de fichiers est installé en se basant sur les attributs de qualification suivants : attribut
		• `drive` permettant de déterminer quel répertoire correspond au système de fichiers installé
		• attribut `nosuid` permettant de déterminer si l'option d'installation est paramétrée si le système de fichiers est installé
			Chaîne présentant le format de qualificatif suivant :
			`[drive:`*`dir_name`*`,`
			`mount_option` `:`
			`false|true]`
			`True|False`
			
			Par exemple, pour vérifier que le répertoire `/home` est monté et que l'option `nosuid` n'est pas définie :
			`os.mountcheck=[drive:/home,`
			`nosuid:false]True`

`os.package.`*`package_name`*	UNIX	Vérifie que la version prise en charge du module *package_name* est installée sur la machine. Chaîne permettant de représenter *package_name*, par exemple, en gras : • Interpréteur de commandes **`bash`** • **`expect`** pour le module d'extension TCL • **`libgcc`** pour le module d'exécution de bas niveau GCC • **`openssh`** pour l'interpréteur de commandes sécurisé Open Source • **`openssl`** pour le kit d'outils Open Source de SSL/TLS • **`perl`** pour le module de script Perl • **`rpm`** pour le RPM ou les modules de génération RPM • **`telnet`** pour le module Telnet • **`wget`** pour le module de récupération de fichiers GNU	Chaîne, par exemple : • `bash-3.2+` for bash shell • `expect-1.2.0` pour Expect • `libgcc-3.4.3-9` pour libgcc • `openssh-5.0.0.5301-` pour openssh • `openssl-4.2.0-` pour OpenSSL • `perl-5.8.2` pour Perl • `rpm` • `telnet` • `wget` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`os.pagesize`	UNIX	Vérifie la taille de page du système.	Format numérique en Ko, par exemple : `4 Ko` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`os.RAMSize`	Tous	Vérifie la mémoire vive du système	Format numérique en Go, par exemple `8 Go`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.SeaMonkeyVersion`	UNIX	Vérifie une version spécifique de Mozilla SeaMonkey sur l'ordinateur étant donné que son chemin est indiqué dans la variable d'environnement `PATH`	Format numérique, par exemple `2.10` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.

`os.SELinux`	Linux	Vérifie le statut d'application de la fonction Linux de sécurité en se basant sur les attributs de qualificatitif suivants :
		• attribut `source` permettant de déterminer la commande à utiliser pour le système d'exploitation associé
			Cette value peut avoir l'un des types suivants :
			• Chaîne présentant le format de qualificatif suivant :
			`[source:Command] Disabled|Enabled`
			Par exemple, pour vérifier que la fonction est désactivée ou présente un statut autorisé sur le système d'exploitation Red Hat ou SUSE :
			`os.SELinux=[source: Command]Disabled`
			• Chaîne sans qualificatif dans laquelle le système d'exploitation est une variante générique Linux :
			`os.SELinux=Disabled`

`os.servicePack`	Tous	Vérifie la version actuelle du module de mise à jour installé	Format numérique, avec *majorVersion*. *minorVersion* ou la version *majorVersion* uniquement Par exemple, pour vérifier que le module de mise à jour 2 ou ultérieur est installé, `2+` **Remarque :** . Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`os.shell.default`	UNIX	Vérifie que le script d'interpréteur de commandes par défaut est installé	Chaîne permettant de représenter le script d'interpréteur de commandes, par exemple les propriétés de conditions prérequises `bash`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.space.dir_name` Prerequisite Scanner comprend trois variantes de la propriété `os.space.dir_name` : • `os.space.dir_name` qui vérifie si l'espace disque disponible est suffisant pour le système de fichiers spécifié, que l'utilisateur connecté soit toujours un superutilisateur ou non. Utilisez cette variante de propriété de prérequis lorsque vous souhaitez vérifier le chemin spécifié du système de fichiers, mais il importe peu que l'utilisateur connecté soit toujours un superutilisateur ou non. **Remarque :** Vous ne pouvez pas utiliser deux fois cette variante pour le même système de fichiers, mais vous pouvez utiliser différents types d'utilisateurs dans un seul fichier de configuration ; utilisez plutôt une combinaison des deux autres variantes. • `os.space.dir_name_nonroot` qui vérifie si l'espace disque disponible est suffisant pour le système de fichiers spécifié de l'utilisateur non superutilisateur. Utilisez cette variante de propriété de prérequis lorsque vous êtes connecté en tant qu'utilisateur non superutilisateur et que vous souhaitez vérifier de manière explicite le chemin spécifié pour le système de fichiers. **Remarque :** L'utilisateur non superutilisateur doit être le même que celui qui installe le produit sur le système cible. • `os.space.dir_name_root` qui vérifie si l'espace disque disponible est suffisant pour le système de fichiers spécifié du superutilisateur. Utilisez cette variante de propriété de prérequis lorsque vous êtes connecté en tant que superutilisateur et que vous souhaitez vérifier de manière explicite le chemin spécifié pour le système de fichiers. Vous pouvez indiquer les variantes `os.space.dir_name_nonroot` et `os.space.dir_name_root` dans le même fichier de configuration. Prerequisite Scanner affiche `NOT_REQ_CHECK_ID` dans la cellule de résultats réels de la variante non applicable. Par exemple, si l'utilisateur connecté est un superutilisateur, Prerequisite Scanner affiche `NOT_REQ_CHECK_ID` pour la variante `os.space.dir_name_nonroot`.			

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides

os.space.*dir_name*	UNIX	Vérifie l'espace disque disponible pour le système de fichiers <*dir_name*> spécifié en se basant sur un ou plusieurs des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au système de fichiers à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
		
		La valeur de l'attribut `dir` dépend de l'utilisateur connecté ; ainsi, la valeur est une paire de valeurs de nom permettant de représenter le type d'utilisateur, à savoir superutilisateur ou non, et le chemin d'accès associé.
		
		dir_name peut représenter par exemple :
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
		
		Remarque : Vous ne pouvez pas utiliser deux fois cette variante pour le même système de fichiers, mais vous pouvez utiliser différents types d'utilisateurs dans un seul fichier de configuration. Utilisez une combinaison des variantes `os.space.`*`dir_name`*`_nonroot` et `os.space.`*`dir_name`*`_root`.
			Chaîne présentant le format de qualifiant suivant pour le système de fichiers d'un superutilisateur :
			`[dir:`**`root`**`=`<*dir_path*>`, unit:`<*unit_name*>`] `<*disk_space*>
			
			Par exemple :
			`os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200`
			
			Chaîne présentant le format de qualificatif suivant pour le système de fichiers d'un utilisateur non superutilisateur :
			`[dir:`**`non_root`**`=`<*dir_path*>`, unit:`<*unit_name*>`] <disk_space>`
			
			Par exemple :
			`os.space.home= [dir:non_root=USERHOME/.acsi_HOST, unit:MB]200`
			
			Chaîne présentant le format de qualificatif suivant utilisant un seul qualificatif :
			`[dir:`<*dir_path*>`] `<*disk_space*> `MB`
			
			Par exemple :
			`os.space.home= [dir:/home/sat]250MB`
			
			Format numérique en Mo ou Go, par exemple :
			`os.space.opt=11GB`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides

os.space.*dir_name*_nonroot	UNIX	Vérifie l'espace disque disponible pour le système de fichiers *dir_name* de l'utilisateur non superutilisateur en se basant sur un ou plusieurs des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au système de fichiers à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
		
		dir_name peut représenter par exemple :
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Chaîne présentant le format de qualificatif suivant pour le système de fichiers d'un utilisateur non superutilisateur :
			`[dir:`**`non_root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] disk_space`
			
			Par exemple :
			`os.space.home_nonroot= [dir:non_root= USERHOME/.acsi_HOST, unit:MB]200`
			
			Chaîne comprenant l'attribut de qualification `dir` uniquement pour le système de fichiers d'un utilisateur non superutilisateur :
			`[dir:`**`non_root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			
			Par exemple :
			`os.space.opt_nonroot= [dir:non_root=/opt/IBM/ITM] 1024MB`

os.space.*dir_name*_root	UNIX	Vérifie l'espace disque disponible pour le système de fichiers *dir_name* du superutilisateur en se basant sur un ou plusieurs des attributs de qualification suivants :
		• attribut `dir` permettant de déterminer le chemin d'accès au système de fichiers à contrôler
		• attribut `unit` permettant de déterminer les unités de l'espace disque à utiliser
		
		dir_name peut représenter par exemple :
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Chaîne présentant le format de qualifiant suivant pour le système de fichiers d'un superutilisateur :
			`[dir:`**`root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			
			Par exemple :
			`os.space.usr_root= [dir:root= /usr/ibm/common/acsi, unit:GB]200`
			
			Chaîne comprenant l'attribut de qualification `dir` uniquement pour le système de fichiers d'un superutilisateur :
			`[dir:`**`root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			
			Par exemple :
			`os.space.opt_root= [dir:root=/opt/IBM/ITM] 1024MB`

`os.sshdConfig`	UNIX	Vérifie que la connexion root autorisée est configurée pour les sessions démon SSH	`Available|Unavailable`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.swapSize`	UNIX	Vérifie que l'espace de permutation doit être supérieur à la taille de la mémoire vive ou à la taille totale de l'espace de permutation	Cette value peut avoir l'un des types suivants : • Valeur booléenne, par exemple : `True` • Format numérique en Mo ou Go, par exemple : `2GB`

`os.tmpdir`	UNIX	Vérifie les autorisations d'accès affectées au système de fichiers `/tmp`, y compris les autorisations spécifiques définies par les indicateurs de droit d'accès, par exemple les bits `sticky`, `setuid` ou `setgid` dans les chiffres octaux	Nombre représentant les chiffres octaux `octal_digits` pour les autorisations d'accès
			Par exemple, pour vérifier que le répertoire temporaire dispose des autorisations `drwxrwxrwt` avec l'autorisation de bit sticky activée :
			`1777`
			Autre exemple, pour vérifier que le répertoire temporaire dispose des autorisations `drwxrwxrwx` excluant le bit sticky :
			`777`

`os.totalMemory`	Windows	Taille totale de mémoire virtuelle à laquelle le système d'exploitation peut accéder	Format numérique en Mo ou Go, par exemple : `4 Go`
`os.totalPhysicalMemory`	Windows	Taille totale de la mémoire physique à laquelle le système d'exploitation peut accéder, mais n'indique pas la taille réelle de la mémoire physique sur l'ordinateur cible	Format numérique en Mo ou Go, par exemple : `2030 Mo`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides

`os.ulimit`	UNIX	Vérifie qu'un nombre illimité de processus peut être exécuté, en se basant sur les attributs de qualification suivants :
		• attribut `type` permettant de déterminer quel attribut supplémentaire contrôler, par exemple `filedescriptorlimit` vérifie la limite du nombre de descripteurs de fichiers que les processus peuvent ouvrir
		
		Il vérifie également que les limites suivantes ont été définies pour les domaines spécifiés dans le fichier `/etc/security/limits.conf` :
		`root - stack unlimited`
		`ctginst1 - stack unlimited`
		`root - nofile 8192`
		`tioadmin - nofile 32767`
			Cette value peut avoir l'un des types suivants :
			• Chaîne présentant le format de qualificatif suivant :
			`[type:`*`limit_name`*`]`
			`limit_value``,`
			`limited|unlimited`
			Par exemple, pour vérifier que la limite du descripteur de fichier est supérieure à 8192, avec un nombre illimité de processus :
			`os.ulimit=[type:`
			`filedescriptorlimit]`
			`8192+,`
			`unlimited`
			
			Voici les types valides de limites à vérifier, où *limit_name* représente le type de limite de la manière suivante :
			– `ALL`, vérifie toutes les limites
			– `corefilesizelimit`
			– `datasegmentlimit`
			– `filedescriptorlimit`
			– `filesizelimit`
			– `hardlimit`
			– `processlimit`
			– `maxmemorysizelimit`
			– `maxprocesseslimit`
			– `stacksizelimit`
			– `threadlimit`
			• `Available|Unavailable` permet d'indiquer si les domaines adéquats comprennent des ensembles de limites dans le fichier `/etc/security/limits.conf`.

`os.umask`	UNIX	Vérifie les autorisations pour le masque de mode de création de fichier	Nombre représentant les chiffres octaux *`octal_digits`* pour les autorisations d'accès. Par exemple, afin de vérifier que les nouveaux fichiers sont uniquement accessibles pour le propriétaire, définissez le chiffre octal sur `0022`
`os.userLimits`	UNIX	Vérifie que l'espace mémoire maximal est illimité ; retourne `Available` le cas échéant	`Available|Unavailable`

Tableau 24. Propriétés de données du système d'exploitation (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`os.versionNumber`	Windows	Vérifie la version actuelle du système d'exploitation installé sur la machine	Format numérique, par exemple `5.0+` **Remarque :** The values can use the special characters as outlined in tableau 1, à la page 2.
`os.windowManager`	UNIX	Checks whether GNOME or KDE is available as a graphical desktop	`Available|Unavailable`

Propriétés de données du logiciel installé

Les propriété de données du logiciel installé vérifient les prérequis du logiciel installé tels que les programmes enregistrés dans le registre Windows et cygwin et gskit sont installés. Pour les systèmes Windows uniquement, il utilise les collecteurs du logiciel installé dans le répertoire *ips_root*`/lib` avec l'identificateur à préfixe `installedSoftware, cygwin` ou `gskit` dans leur nom de fichier.

tableau 25 présente les propriétés de prérequis de données communes. Cette catégorie de propriétés de prérequis n'exige pas d'identificateur à préfixe.

Tableau 25. Propriétés de données du logiciel installé

Propriété de prérequis	Plateforme	Description	Valeurs valides
`installedSoftware`	Windows	Recherche dans le registre du système d'exploitation des programmes installés avec leurs emplacements	Chaîne avec plusieurs applications séparées par une virgule.
`cygwinVersion`	Windows	Vérifie la version de cygwin installée sur la machine ; renvoie 0.0 si aucune version n'est installée	Entier positif, par exemple `1,5` **Remarque :** Les valeurs peuvent utiliser les caractères spéciaux comme indiqué dans tableau 1, à la page 2.
`gskit7Version`	Windows	Vérifie que la version 7 de gskit est installée sur la machine ; renvoie 0.0 si ce n'est pas le cas	Entier positif, par exemple `7,0`
`gskit8Version`	Windows	Vérifie que la version 8 de gskit est installée sur la machine ; renvoie 0.0 si ce n'est pas le cas	Entier positif, par exemple `8,0`

Propriétés de données utilisateur

Les propriétés de données utilisateur vérifient les prérequis de l'utilisateur, par exemple que l'utilisateur connecté dispose des droits d'administration ou est le superutilisateur. Pour les systèmes Windows uniquement, il utilise le collecteur utilisateur dans le répertoire *ips_root*`/lib` avec l'identificateur à préfixe `user` dans leur nom de fichier. Pour les systèmes UNIX uniquement, il utilise le collecteur utilisateur dans le répertoire *ips_root*`/lib/packageTest.sh`.

tableau 26, à la page 114 présente les propriétés de prérequis utilisateur. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `user`.

Tableau 26. Propriétés de données utilisateur

Propriété de prérequis	Plateforme	Description	Valeurs valides
`user.userID`	Windows	ID de l'utilisateur actuellement connecté	Chaîne, par exemple `smithj`
`user.isAdmin`	Tous	Vérifie que l'utilisateur connecté est membre du groupe administrateur	Valeur booléenne, par exemple `True`

Propriétés de données du réseau Windows

Les propriétés de données du réseau Windows vérifient les prérequis du réseau, par exemple si NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING. Le système utilise les collecteurs de réseau Windows dans le répertoire *ips_root*`/lib` avec l'identificateur à préfixe `network` dans leur nom de fichier. .

tableau 27 présente les propriétés de prérequis de réseau communes à toutes les plateformes Windows. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `network`.

Tableau 27. Propriétés de données du réseau Windows

Propriété de prérequis	Description	Valeurs valides
`network.DHCPEnabled`	Vérifie qu'au moins un adaptateur d'une adresse IP valide a obtenu cette adresse IP à l'aide de DHCP ; renvoie `True` s'il y en a au moins un.	Valeur booléenne, par exemple `False`
`network.netBIOSEnabled`	Vérifie qu'au moins un adaptateur doté d'une adresse IP valide dispose de NetBIOS activé comme protocole ; renvoie `True` s'il y en a au moins un.	Valeur booléenne, par exemple `True`
`network.pingLocalhost`	Vérifie que le système hôte local répond au protocole de la commande PING ; retourne `True` le cas échéant.	Valeur booléenne, par exemple `True`
`network.pingSelf`	Vérifie que le nom de l'ordinateur local a été trouvé à l'aide de DCHP et qu'il est possible de lui soumettre la commande PING ; renvoie `True` le cas échéant.	Valeur booléenne, par exemple `True`
`network.ValidateHostsFile`	Vérifie que les entrées de `C:\WINDOWS\system32\drivers\etc\hosts` se présentent au format correct ; retourne `True` si le format est valide.	Valeur booléenne, par exemple `True`

Propriétés de données du réseau UNIX

Les propriétés de données du réseau UNIX vérifient les prérequis du réseau, par exemple que NetBIOS et DHCP sont activés sur la machine, ainsi que les propriétés de la commande PING. Le système utilise les collecteurs de réseau dans le répertoire *ips_root*`/UNIX_Linux`.

tableau 28 présente les propriétés de prérequis de réseau communes à toutes les plateformes UNIX. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `network`.

Tableau 28. Propriétés de données du réseau UNIX

Propriété de prérequis	Description	Valeurs valides
`network.DHCPEnabled`	Vérifie qu'au moins un adaptateur doté d'une adresse IP valide a obtenu cette adresse IP à l'aide de DHCP.	Valeur booléenne, par exemple `False`

Tableau 28. Propriétés de données du réseau UNIX (suite)

Propriété de prérequis	Description	Valeurs valides
`network.dns`	Vérifie que l'entrée DNS de la machine hôte est correcte	Valeur booléenne, par exemple `True`
`network.fqdn`	Vérifie que le nom de domaine complet de la machine hôte est défini	Valeur booléenne, par exemple `True`
`network.pingLocalhost`	Vérifie que le système hôte local répond au protocole de la commande PING.	Valeur booléenne, par exemple `True`
`network.pingSelf`	Vérifie que le nom de l'ordinateur local a été résolu à l'aide de DCHP et qu'il est possible de lui soumettre la commande PING.	Valeur booléenne, par exemple `True`

Propriétés de données de la variable d'environnement

Les propriétés de données de la variable d'environnement vérifient les prérequis de variables d'environnement pouvant être communs à toutes les plateformes, comme la définition ou non d'une variable d'environnement ou sa valeur. Pour les systèmes Windows uniquement, le système utilise les collecteurs dans le répertoire *ips_root*`/lib/`, avec l'identificateur à préfixe `env` figurant dans leur nom de fichier. Pour les systèmes UNIX uniquement, le système utilise les collecteurs de variable d'environnement UNIX dans le répertoire *ips_root*`/UNIX_Linux` avec l'identificateur à préfixe `env` figurant dans leur nom de fichier.

tableau 29 présente brièvement les propriétés de prérequis de variable d'environnement communes à toutes les plateformes. Cette catégorie de propriétés de prérequis exige l'identificateur à préfixe `env`.

Tableau 29. Propriétés de données de la variable d'environnement

Propriété de prérequis	Plateforme	Description	Valeurs valides
`env.var.set.` *`env_var_name`*	UNIX	Utilisez cette convention d'attribution de nom pour vérifier que la variable d'environnement *env_var_name* indiquée est définie sur l'ordinateur, par exemple : • `env.var.set.HOME` vérifie que la variable d'environnement est définie pour le répertoire de base, où *env_var_name* est le nom de la variable d'environnement `HOME` • `env.var.set.JAVA_HOME` vérifie que la variable d'environnement est définie pour le répertoire de base de Java, où *env_var_name* est le nom de la variable d'environnement `JAVA_HOME`	Valeur booléenne, par exemple `True`

Tableau 29. Propriétés de données de la variable d'environnement (suite)

Propriété de prérequis	Plateforme	Description	Valeurs valides
`env.var.set.`*`env_var_name`* `[type:`*`env_var_type`*`]`	Windows	Utilisez cette convention d'attribution de nom pour vérifier que la variable d'environnement *env_var_name* indiquée est définie pour le type de variable d'environnement *env_var_type* indiqué, par exemple : • `env.var.set.HOME` vérifie que la variable d'environnement est définie pour le répertoire de base, où *env_var_name* est le nom de la variable d'environnement `HOME` • `env.var.set.JAVA_HOME[type:User]` vérifie que la variable d'environnement du répertoire de base de Java est définie pour l'utilisateur enregistré, où *env_var_name* est le nom de la variable d'environnement `JAVA_HOME` et *env_var_type* est le type de variable d'environnement `User` Le type de variable d'environnement *env_var_type* est facultatif et représente les types de variables d'environnement pris en charge par le système d'exploitation Windows de la manière suivante : • `Process` • `System` • `User` • `Volatile` Si non indiqué, le type par défaut est `Process`.	Valeur booléenne, par exemple `True`
`env.classpath.derbyJAR`	Tous	Vérifiez si le chemin d'accès au fichier Derby JAR file est contenu dans la variable d'environnement du chemin d'accès aux classes	Valeur booléenne, par exemple `True`
`env.CIT.homeExists`	Windows	Vérifiez que les deux variables d'environnement `HOMEDRIVE` et `HOMEPATH` existent	Valeur booléenne, par exemple `True`

Annexe D. Collecteurs prédéfinis pour systèmes UNIX

Il existe des collecteurs individuels pour des propriétés de prérequis contenus dans des systèmes UNIX qui sont dans le *ips_root*/répertoire lib. Vous pouvez revoir ces collecteurs et leurs paramètres d'entrée avant de créer des collecteurs personnalisés.

tableau 30 décrit les collecteurs prédéfinis pour les systèmes UNIX.

Tableau 30. collecteurs UNIX

collecteur	pour une propriété de prérequis	entrées
DB2_Version	DB2 Version	aucun
DBType	DBType	aucun
DBTypeDetails	DBTypeDetails	aucun
env.classpath.derbyJAR	env.classpath.derbyJAR	aucun
env.var.set	env.var.set*env_var_name* *env_var_name* est le nom de la variable d'environnement à vérifier	*$env_var_name*
network.DHCPEnabled	network.DHCPEnabled	aucun
network.dns	network.dns	aucun
network.fqdn	network.fqdn	aucun
network.pingSelf	network.pingSelf	aucun
network.port	network.availablePorts.* network.portsInUse.*	$ports
oracle.Client	oracle.Client	aucun
oracle.Client.Location	oracle.Client.Location	aucun
oracle.Server	oracle.Server	aucun
oracle.Server.Location	oracle.Server.Location	aucun
os.architecture	os.architecture	32 bit|64 bit
os.automount	os.automount	aucun
os.cmd	os.lookup	nslookup
os.cmd	os.tar|os.gnu.tar	tar|gtar
os.dir	os.dir.*dir_name* *le nom_répertoire* peur représenter par exemple: • tmp • accueil	Chaîne dans le format suivant: [dir:*dir_name*, type:permission] *octal_digits+* Par exemple, pour vérifier si le *dir_name* répertoire, c'est-à-dire, le répertoire d'accueil dispose de drwxr-xr-x permissions: [dir:/home, type:permission]755+
os.diskquota	os.diskquota	aucun
os.expectLink	os.expectLink	aucun

Tableau 30. collecteurs UNIX (suite)

collecteur	pour une propriété de prérequis	entrées
os.filepath	os.file.*script_name* *script_name* peut représenter par exemple: • bash • attendre • gzip • archivage sur bande	chemin d'accès au fichier script, où le chemin d'accès peut être: • /bin/bash • /usr/bin/expect • /usr/bin/gzip • /usr/bin/tar
os.Firefox	os.Firefox	aucun
os.FreePagingSpace	os.FreePagingSpace	aucun
os.ftpusers	os.ftpusers	aucun
os.hostformat	os.hostformat	aucun
os.iodevicestatus	os.iodevicestatus	aucun
os.kernelMode	os.kernelMode	aucun
os.kernelParameters	os.kernelParameters	aucun
os.kernelversion	os.kernelversion	aucun
os.largeFile	os.largeFile	aucun
os.ldLibPath	os.ldLibPath	produit
os.level	os.level	aucun

Tableau 30. collecteurs UNIX (suite)

collecteur	pour une propriété de prérequis	entrées
os.lib	os.lib.*lib_name_version* Chaîne ou expression régulière pour représenter *lib_name_version*, par exemple, en gras: • 32-bit **libstdc++.so.#** bibliothèque • 64-bit **libstdc++.so.#** bibliothèque • 32-bit **libXft.so.#** bibliothèque • 32-bit **libXtst.so.#** bibliothèque • 64-bit **libaio.so.#** bibliothèque • 32-bit **xlC.rte** niveau d'exécution XLC • 32-bit **xlC.aix50.rte** exécution XLC pour la version 5.3 AIX • 32-bit **xlC.aix61.rte** exécution XLC pour la version 6.1 AIX • AIX IOCP **bos.iocp.rte** bibliothèque • **bos.loc.iso.en_us**, l'ensemble de fichiers code ISO pour AIX base operating system • **regex{*str*}**, une expression régulière avec le paramètre d'entrée,*str*, représentant le motif recherché pour le nom de bibliothèque, par exemple, .*libgcc.*	*path_to_library* ou *nom de version lib* par exemple, pour vérifier la valeur réelle de os.lib.libstdc++.so la propriété de prérequis, les paramètres d'entrée sont /usr/lib/libstdc++.so.5 et libstdc++.so: os.lib /usr/lib/libstdc++.so.5 libstdc++.so Par exemple pour vérifier si une version de la bibliothèque d'exécution de bas niveau GCC, libgcc, s'affiche sur la machine, le paramètre d'entrée est: regex{.*libgcc.*}
os.loginVariable	os.loginVariable	aucun
os.maximoDirectory	os.maximoDirectory	aucun
os.maximoDirOwner	os.maximoDirOwner	aucun
os.maximumProcesses	os.maximumProcesses	aucun
os.MozillaVersion	os.MozillaVersion	aucun
os.mountcheck	os.mountcheck	Chaîne avec le format de qualificateur suivant: [unité:*dir_name*, *mount_option* : faux|vrai] vrai|Faux par exemple, pour vérifier si /le répertoire d'accueil est monté et nosuid l'option n'est pas définie: os.mountcheck=[drive:/home, nosuid:faux]vrai

Tableau 30. collecteurs UNIX (suite)

collecteur	pour une propriété de prérequis	entrées
`os.package`	`os.package.`*`package_name`* Chaîne pour représenter *package_name*, par exemple, en gras: • **`bash`** shell • **`expect`** pour l'extension de module TCL • **`libgcc`** pour le module d'exécution de bas niveau GCC • **`openssh`** pour connexion secure shell de Open Source • **`openssl`** pour la boîte à outils d'Open Source pour SSL/TLS • **`perl`** pour le module Perl scripting • **`rpm`** pour la construction de modules RPM ou RPM • **`telnet`** pour le module Telnet • **`wget`** pour le module de récupération du fichier GNU	*package_name*, par exemple, où *package_name* est `rpm`: `module.os rpm`
`os.pagesize`	os.pagesize	aucun
`os.RAMSize`	os.RAMSize	aucun

`os.SELinux`	`os.SELinux`	• Chaîne avec le format de qualificateur suivant:
		`[source:Command] Disabled|Enabled`
		par exemple, pour vérifier si le dispositif est désactivé ou a un statut permissif sur l'un des systèmes d'exploitation Red Hat ou SUSE:
		`os.SELinux=[source: Command]Disabled`
		• S'il n'existe pas de qualificateur, aucune valeur n'est passée au collecteur.

`os.servicePack`	`os.servicePack`	Service pack à valeur
`os.shell.default`	`os.shell.default`	La valeur attendue de la propriété de prérequis, par exemple, `bash`

Tableau 30. collecteurs UNIX (suite)

collecteur	pour une propriété de prérequis	entrées
os.space	os.space.*dir_name* *le nom_répertoire* peur représenter par exemple: • usr • accueil • tmp • var	Chaîne avec le format de qualificateur suivant pour le système de fichier du superutilisateur : [dir:**root**=*dir_path*, unité :*unit_name*] *disk_space* Par exemple : os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200 Chaîne avec le format de qualificateur suivant pour le système de fichier d'un non superutilisateur: [dir:**non_root**=*dir_path*, unité :*unit_name*] espace disque par exemple: espace d'accueil.os= [dir:non_root=USERHOME/ .acsi_HOST, unité:MB]200 Chaîne avec le format de qualificateur suivant, utiliser un seul qualificateur : [dir:*dir_path*] *disk_space* MB par exemple: espace d'accueil.os= [dir:/home/sat]250MB
os.sshdConfig	os.sshdConfig	aucun
os.swapSize	os.swapSize	aucun
os.tmpdir	os.tmpdir	aucun

Tableau 30. collecteurs UNIX (suite)

collecteur	pour une propriété de prérequis	entrées
os.ulimit	os.ulimit	Chaîne avec le format de qualificateur suivant: [type:*limit_name*] *limit_value*, limité|illimité par exemple, vérifier si le descripteur de fichier limité est supérieur à 8192, avec des processus de nombres illimités : os.ulimit= [type:filedescriptorlimit] 8192+,unlimited Validez les types de limites pour vérifier, où *nom de limite* représente le type de limite comme suit : • ALL, vérifie toutes les limites • corefilesizelimit • datasegmentlimit • filedescriptorlimit • filesizelimit • hardlimit • processlimit • maxmemorysizelimit • maxprocesseslimit • stacksizelimit • threadlimit
os.umask	os.umask	aucun
os.userLimits	os.userLimits	aucun
os.windowManager	os.windowManager	aucun

Annexe E. Fonctions communes pour les systèmes Windows

Prerequisite Scanner possède un ensemble de fonctions communes dans le fichier `/lib/common_function.vbs` pour l'exécution de vérifications sur les systèmes Windows.

Tableau 31. Fonctions dans common_function.vbs

Fonction	Description
«allFiles()», à la page 124	Lit les noms de fichiers dans un répertoire spécifique à l'intérieur d'un tableau.
«arrayToString()», à la page 125	Crée une représentation de chaîne pour la matrice.
«bigthan()», à la page 125	Calcule la différence entre la valeur attendue et la valeur réelle de la propriété de prérequis si cette dernière a une taille en Mo/s ou Go/s.
«changeMG()», à la page 126	Convertit le paramètre d'entrée en Mo/s ou Go/s pour les propriétés de prérequis d'espace disque ou de mémoire.
«checkItemToString()», à la page 126	Crée une représentation de chaîne pour l'objet `CheckItem`.
«dictionaryToString()», à la page 127	Crée une représentation de chaîne pour l'objet dictionnaire de scriptage.
«exeCommand()», à la page 127	Exécute la commande spécifiée et retourne le résultat à partir de l'exécution de cette commande.
«filterCommand()», à la page 127	Exécute la commande spécifiée et retourne les lignes à partir du résultat de la commande qui correspond au modèle spécifié.
«filterFile()», à la page 128	Lit et filtre le contenu d'un fichier à un objet de scriptage dictionnaire.
«findNewest()», à la page 129	Recherche le fichier de configuration le plus récent.
«findSuitableFile()», à la page 129	Recherche le fichier de configuration adéquat pour un produit et une version.
«fmt()», à la page 130	Modifie une chaîne en y ajoutant un nombre défini de caractères issus d'une autre chaîne et en remplissant l'autre chaîne de caractères d'espace si la longueur de l'autre chaîne est insuffisante ou en tronquant l'autre chaîne si celle-ci est trop longue.
«formatForDisplay()», à la page 131	Met en forme le paramètre d'entrée pour le rendre lisible.
«formatSizeForDisplay()», à la page 131	Prend le paramètre d'entrée et ajoute ou réduit la partie décimale du paramètre d'entrée à deux séparateurs décimaux, par exemple, 123 Mo à 123,00 Mo ou 12,123 Mo/s à 12,12 Mo/s.

Tableau 31. Fonctions dans common_function.vbs (suite)

Fonction	Description
«getDecimalSeparator()», à la page 131	Détermine le séparateur décimal qui est utilisé pour l'environnement local en cours.
«getFirstMatch()», à la page 132	Obtient la première correspondance de la chaîne de recherche dans la matrice.
«isMatch()», à la page 132	Vérifie si le modèle de recherche se trouve dans la chaîne.
«pas en dernier ()», à la page 133	Filtre la première matrice pour déterminer si le contenu se trouve dans la seconde matrice. En fonction de la valeur du paramètre d'entrée `in_or_not`, la fonction retourne les contenus de la première matrice en incluant ou excluant ceux qui correspondent à la seconde matrice.
«succès ou échec ()», à la page 133	Compare les valeurs attendue et réelle de la propriété de prérequis et détermine si la propriété de prérequis réussit le contrôle. Les paramètres d'entrée peuvent être des nombres génériques, une taille en Mo/s ou Go/s, une vitesse d'unité centrale en MHz ou GHz, une valeur booléenne, ou des chaînes.
«ppread()», à la page 134	Lit les contenus d'un fichier sur un objet dictionnaire de scriptage, en fractionnant davantage chaque ligne du fichier via le paramètre d'entrée du séparateur spécifié si ce séparateur existe dans la ligne.
«readFile()», à la page 135	Lit chaque ligne d'un fichier dans une entrée d'index d'une matrice.
«unitMGTOG()», à la page 135	Rassemble les contenus d'une matrice pour obtenir le nombre total de Mo/s.
«varToString()», à la page 135	Crée une représentation sous forme de chaîne d'une variable. La variable à vérifier peut être une chaîne, un nombre, un objet dictionnaire de scriptage, une matrice, ou un objet `CheckItem`.

allFiles()

Lit les noms de fichiers d'un répertoire spécifié dans une matrice.

Objectif

Cette fonction obtient la liste des fichiers dans le paramètre d'entrée du répertoire et ajoute ces derniers à la matrice. Elle retourne la matrice.

Syntaxe

```
allFiles(filepath)
```

Paramètres d'entrée

Chaîne *filepath*
Le chemin d'accès au répertoire qui contient les fichiers.

Valeurs de retour

Matrice *fileNames*
Retourne la matrice contenant les noms de fichiers dans le répertoire spécifié.

arrayToString()

Crée une représentation de chaîne pour la matrice.

Objectif

Cette fonction prend la matrice qui est transmise comme paramètre d'entrée et retourne une représentation sous forme de chaîne des contenus de cette matrice.

Syntaxe

```
arrayToString(arr)
```

Paramètres d'entrée

Matrice *arr*
Contient la matrice.

Valeurs de retour

Chaîne *résultat*
Retourne une représentation sous forme de chaîne de la matrice, avec chaque élément séparé par une virgule.

bigthan()

Calcule la différence entre la valeur attendue et la valeur réelle de la propriété de prérequis si cette dernière a une taille en Mo/s ou Go/s.

Objectif

Cette fonction appelle d'abord la fonction «changeMG()», à la page 126 pour changer les valeurs attendue et réelle de la propriété de prérequis en Mo/s si besoin. Elle vérifie ensuite si l'une ou l'autre des valeurs retournées depuis les fonctions est "null", et si tel est le cas, la valeur retournée de la fonction est 0 Mo et la fonction existe. Elle vérifie si l'une des valeurs est en Mo ou Go, et convertit en Mo/s si besoin. Elle calcule la différence entre les valeurs formatées finales et retourne le résultat.

Syntaxe

```
bigthan(expect,real)
```

Paramètres d'entrée

Chaîne *expect*
La valeur attendue de la propriété de prérequis.

Chaîne *real*
La valeur réelle de la propriété de prérequis.

Valeurs de retour

Chaîne *bigthan*
Retourne la différence en Mo/s ou 0 Mo/s s'il n'y a aucune différence.

changeMG()

Met en forme le paramètre d'entrée pour en supprimer les éventuels caractères de groupement de caractères numériques supplémentaires et retourne le paramètre mis en forme sauf si le paramètre d'entrée contient des Mo/s ou Go/s. Si tel est le cas, cette fonction convertit le paramètre d'entrée en Go/s ou Mo/s respectivement.

Objectif

Cette fonction appelle la fonction «getDecimalSeparator()», à la page 131 pour déterminer le séparateur décimal de l'environnement local puis supprime les éventuels caractères de groupement de caractères numériques supplémentaires pour cet environnement local depuis le paramètre d'entrée de nombre. Elle appelle ensuite la fonction «getFirstMatch()», à la page 132 pour déterminer si la valeur est en Mo/s ou Go/s puis convertit cette valeur en Go/s ou Mo/s respectivement.

Syntaxe

```
changeMG(tochange)
```

Paramètres d'entrée

Chaîne *tochange*
Contient le format de valeur et convertit le cas échéant.

Valeurs de retour

Chaîne *changeMG*
Retourne soit le nombre mis en forme sans les caractères de groupement de caractères numériques soit le nombre en Mo/s ou Go/s.

checkItemToString()

Crée une représentation de chaîne pour l'objet `CheckItem`.

Objectif

Cette fonction prend l'objet `CheckItem` qui est transmis comme paramètre d'entrée et retourne une représentation sous forme de chaîne qui comprend les valeurs de différentes propriétés pour cette instance de l'objet `CheckItem`.

Syntaxe

```
checkItemToString(var)
```

Paramètres d'entrée

CheckItem *var*
Contient l'instance de l'objet `CheckItem`.

Valeurs de retour

Chaîne *result*
Retourne une représentation sous forme de chaîne pour les propriétés de l'objet `CheckItem` comme suit :

```
result = "CheckItem[pdCode[" & chkItem.pdCode & "],pdName[" & chkItem.pdName & _
         "],itype[" & chkItem.itype & "],recommended[" & chkItem.recommended & _
         "],realValue[" & chkItem.realValue & "],passOrFail[" &
     chkItem.passOrFail & "]]"
```

dictionaryToString()

Crée une représentation de chaîne pour l'objet dictionnaire de scriptage.

Objectif

Cette fonction prend l'objet dictionnaire qui est transmis comme paramètre d'entrée et retourne une représentation sous forme de chaîne des contenus de cet objet de dictionnaire.

Syntaxe

```
dictionaryToString(dic)
```

Paramètres d'entrée

Dictionnaire *dic*
Contient l'objet dictionnaire.

Valeurs de retour

Chaîne *result*
Retourne une représentation sous forme de chaîne de l'objet dictionnaire, avec chaque clé et élément séparé par des symboles "égale".

exeCommand()

Exécute la commande spécifiée et retourne le résultat à partir de l'exécution de cette commande.

Objectif

Cette fonction exécute le paramètre d'entrée de commande. En cas d'erreur, elle appelle la sous-routine `logWarning` pour afficher les erreurs ; sinon, elle retourne le résultat à partir de l'exécution de cette commande.

Syntaxe

```
exeCommand(cmd)
```

Paramètres d'entrée

Chaîne *cmd*
Le nom de la commande à exécuter.

Valeurs de retour

Chaîne *result*
Retourne une chaîne qui contient le résultat à partir de l'exécution de cette commande.

filterCommand()

Exécute la commande spécifiée et retourne les lignes à partir du résultat de la commande qui correspond au modèle spécifié.

Objectif

Cette fonction exécute le paramètre d'entrée de commande. Elle analyse le résultat à partir de l'exécution de cette commande et vérifie si une ligne du résultat correspond au paramètre d'entrée du modèle de ligne. En cas de correspondance, elle appelle la fonction «getFirstMatch()», à la page 132 pour déterminer s'il existe également une correspondance entre le paramètre d'entrée de la ligne d'information et le résultat de la commande. Si tel est le cas, elle utilise la fonction `Join` pour retourner les contenus de l'objet dictionnaire à partir de la fonction `getFirstMatch()`.

Syntaxe

```
filterCommand(cmd, line_patt, after_line, info_patt)
```

Paramètres d'entrée

Chaîne *cmd*
Le nom de la commande à exécuter.

Chaîne *line_patt*
Le modèle de ligne à rechercher dans le résultat à partir de l'exécution de cette commande.

Nombre *after_line*
Le nombre de lignes après lequel la recherche du modèle d'informations s'arrête.

Chaîne *info_patt*
Le modèle d'informations à rechercher dans chaque ligne à partir du résultat de la commande.

Valeurs de retour

Chaîne *filterCommand*
Retourne les contenus de l'objet dictionnaire sous forme d'une chaîne unique.

filterFile()

Lit et filtre les contenus d'un fichier dans un objet dictionnaire de scriptage.

Objectif

Cette fonction lit chaque ligne du fichier et transmet chaque ligne avec le modèle de recherche à la fonction «getFirstMatch()», à la page 132. Si elle retourne une correspondance et que la ligne n'existe pas encore dans l'objet dictionnaire, la ligne est écrite dans l'objet dictionnaire. Cette fonction fait une boucle jusqu'à ce que la fin du fichier soit atteinte puis retourne l'objet dictionnaire.

Syntaxe

```
filterFile(fileName, patt)
```

Paramètres d'entrée

Chaîne *nom de fichier*
Le fichier à filtrer.

Chaîne *patt*
Le modèle à rechercher dans chaque ligne du fichier.

Valeurs de retour

Dictionnaire *dic.keys*
Retourne l'objet dictionnaire *dic* avec les lignes filtrées à partir du fichier.

findNewest()

Recherche le fichier de configuration le plus récent dans une matrice.

Objectif

Cette fonction fait une boucle à travers la matrice et identifie le fichier dans la matrice qui est le fichier de configuration le plus récent. Elle retourne le nom du fichier.

Syntaxe

```
findNewest(arr)
```

Paramètres d'entrée

Matrice *arr*
Contient l'ensemble des fichiers de configuration à vérifier.

Valeurs de retour

Chaîne *result*
Retourne le nom du fichier de configuration le plus récent.

findSuitableFile()

Permet de rechercher le fichier de configuration pertinent d'un produit et d'une version.

Objectif

Cette fonction appelle la fonction «getFirstMatch()», à la page 132 pour obtenir l'ensemble des fichiers dont le paramètre d'entrée d'extension représente l'extension du fichier de la liste des fichiers renvoyés par la fonction «allFiles()», à la page 124. Elle rappelle ensuite la fonction «getFirstMatch()», à la page 132 pour revenir sur l'ensemble des fichiers contenant le paramètre d'entrée du code produit dans le nom du fichier. Elle appelle la même fonction pour obtenir l'ensemble des fichiers contenant le paramètre d'entrée de version dans le nom du fichier. Si la fonction trouve une ou plusieurs correspondances de fichiers pour la version, elle appelle la fonction «findNewest()» pour obtenir la dernière version de ce fichier et renvoie le nom de fichier, sinon, elle renvoie le fichier `common.bat` ou utilise les sous-routines `logScreen` et `logWarning` avant de revenir à la dernière version du fichier de configuration du code produit.

Syntaxe

```
findSuitableFile(pd,version,suf,filepath)
```

Paramètres d'entrée

Chaîne *pd*
Le code produit associé au fichier à rechercher, tel que spécifié dans le fichier du code produit est le fichier *ips_root*`/codename.cfg`.

Chaîne *version*
La version du produit associée au fichier à rechercher. <*version*> est le code à 8 chiffres représentant la version, l'édition, la modification et le niveau, avec deux chiffres pour chaque partie du code, par exemple, 7.3.21 désigne 07032100.

Chaîne *suf*
L'extension du type de fichier à rechercher, comme `cfg` ou `bat`.

chaîne *filepath*
Le chemin d'accès au répertoire contenant le fichier à rechercher.

Valeurs de retour

Chaîne *findSuitableFile*
Renvoie l'un des noms de fichiers suivants en fonction des résultats des fonctions appelées :

- *pd_version*`.cfg`, la dernière version du fichier pour le code produit et la version associés.
- `common.bat` si la valeur du paramètre d'entrée d'extension de fichier est `bat`.
- *pd*`.cfg`, la dernière version du fichier de configuration générique du produit, si aucun fichier contenant le paramètre d'entrée de version n'est disponible.

fmt()

Permet de modifier une chaîne en y ajoutant un nombre spécifié de caractères à partir d'une autre chaîne et en remplissant l'autre chaîne avec des caractères espace si la longueur de l'autre chaîne est trop courte ou en tronquant l'autre chaîne, si elle est trop longue.

Objectif

Cette fonction recherche l'expression `%#s` à l'intérieur du paramètre d'entrée *s* du type chaîne. L'expression `%#s` détermine le nombre spécifié de caractères # à partir du paramètre d'entrée *args* qui sont ajoutés à la première chaîne à la position de cette expression. Si le caractère spécifiée # est supérieur à la longueur du paramètre d'entrée *args*, la différence est rembourrée par des caractères espace. Si le caractère spécifiée # est inférieur à la longueur du paramètre d'entrée *args*, la longueur est tronquée par la différence. Si le caractère spécifiée # est égal à 0, la longueur totale du paramètre d'entrée *args* est ajoutée à la première chaîne, à la position appropriée.

Syntaxe

```
fmt(s, args)
```

Paramètres d'entrée

Chaîne *s*
Contient la chaîne à remplacer par le nombre spécifié de caractères # dans l'expression `%#s` à l'intérieur de cette chaîne.

Tableau *args*
Contient le jeu de caractères qui modifient le paramètre d'entrée *s*.

Valeurs de retour

Chaîne *result*
Permet de renvoyer la chaîne modifiée.

Exemple

```
fmt("Hello %5s!",array("Neo")) renvoie "Hello Neo  !" rembourré par des caractères espace supplémentaires
fmt("Hello %5s!",array("Mr. Anderson")) renvoie "Hello Mr. A!" tronqué pour n'ajouter que "Mr. A"
fmt("Hello %0s!",array("Mr. Anderson")) renvoie "Hello Mr. Anderson!"
```

formatForDisplay()

Permet de mettre en forme le paramètre d'entrée afin qu'il soit lisible.

Objectif

Cette fonction appelle la fonction «formatSizeForDisplay()» pour mettre en forme le paramètre d'entrée.

Syntaxe

```
formatForDisplay(val)
```

Paramètres d'entrée

Variable *val*
La variable à mettre en forme.

Valeurs de retour

Chaîne *vartoString*
Renvoie le résultat de la fonction «formatSizeForDisplay()» appelée.

formatSizeForDisplay()

Ramène le paramètre d'entrée et ajoute ou enlève les espaces de début et de fin de la partie décimale duparamètre d'entrée à deux séparateurs décimaux, par exemple, de 123 Mo à 123.00 Mo ou de 12.123 Mo à 12.12 Mo.

Objectif

Cette fonction permet de compter le nombre de caractères dans le paramètre d'entrée, vérifie s'il s'agit d'un nombre ou d'une chaîne et fractionne le paramètre d'entrée dans les parties entières et décimales. Selon la partie décimale, elle ajoute ou enlève les espaces de début et de fin sur deux positions décimales. Il renvoie le résultat.

Syntaxe

```
formatSizeForDisplay(size)
```

Paramètres d'entrée

Entier *size*
La valeur à arrondir à deux séparateurs décimaux.

Valeurs de retour

Entier *val*
Renvoie la valeur arrondie à deux séparateurs décimaux.

getDecimalSeparator()

Détermine le séparateur décimal utilisé pour les paramètres régionaux.

Objectif

Cette fonction crée un nombre fractionnaire, puis utilise la fonction `Mid()` pour déterminer le séparateur décimal utilisé dans ce nombre fractionnaire.

Syntaxe

```
getDecimalSeparator()
```

Paramètres d'entrée

Aucun

Valeurs de retour

Caractère *sep*
: Renvoie le séparateur décimal, par exemple , ou . pour les paramètres régionaux.

getFirstMatch()

Permet d'obtenir la première correspondance de la chaîne de recherche du tableau.

Objectif

Cette fonction utilise une expression régulière pour rechercher le motif, considéré comme un paramètre d'entrée, dans le tableau qui est également considéré comme un paramètre d'entrée. Lorsqu'elle trouve la première correspondance du motif dans le tableau, elle ajoute la valeur à partir du tableau de l'objet dictionnaire de scriptage.

Syntaxe

```
getFirstMatch(patt, arr)
```

Paramètres d'entrée

Chaîne *patt*
: Contient le motif à rechercher.

Tableau *arr*
: Contient le tableau dans lequel se trouve le motif recherché.

Valeurs de retour

Dictionnaire *keys*
: Renvoie les clés de l'objet dictionnaire de scriptage.

isMatch()

Permet de vérifier si le motif recherché se trouve dans la chaîne.

Objectif

Cette fonction appelle la fonction «getFirstMatch()», en transmettant le motif et la chaîne (contenus dans ce tableau) en tant que paramètres d'entrée à cette fonction. Elle appelle la fonction `ubound` pour vérifier si la valeur retournée depuis la fonction `getFirstMatch()` est supérieure ou égale à 0. Si tel est le cas, il y a une correspondance, sinon il y en a pas.

Syntaxe

`isMatch(patt,str)`

Paramètres d'entrée

Chaîne *patt*
Contient le motif à rechercher.

Chaîne *str*
Contient la chaîne dans laquelle se trouve le motif recherché.

Valeurs de retour

Boolean *True|False*
Renvoie la valeur `True` s'il y a une correspondance, sinon elle renvoie `False`.

pas en dernier ()

Filtre le premier tableau afin de déterminer si le contenu est dans le second tableau. En fonction de la valeur de `dans_ou_ pas` paramètre d'entrée, la fonction retourne le contenu du premier tableau en incluant ou en excluant ce qui correspond au second tableau.

Objectif

Syntaxe

`notInLatter(arr1, arr2, in_or_not)`

Paramètres d'entrée

mat *tableau 1*
copier à partir d'un emplacement

mat *tableau 2*
vers un autre emplacement.

chaîne *interne_ou_externe*
Contient soit `"dans"` soit `"pas"` selon que la fonction doit renvoyer les contenus du premier tableau filtré pour ne renvoyer que les contenus qui correspondent au second tableau (`"dans"`) ou les contenus qui ne correspondent pas au second tableau (`"not"`).

Valeurs de retour

Dictionnaire *clé*
Renvoie les clés de l'objet dictionnaire de scriptage qui contient le premier tableau filtrée pour n'avoir que les contenus qui correspondent au second tableau (`dans_ou_pas = "dans"`) ou les contenus qui ne correspondent pas au second tableau ou (`dans_ou_pas = "pas"`).

succès ou échec ()

Compare les valeurs attendues et les valeurs réelles de la propriété de prérequis et détermine si celle-ci réussit à l'épreuve de vérification. Les numéros d'entrée peuvent être des numéros génériques, en Mb ou Gb, vitesse CPU en MHz ou GHz, booléen, ou chaînes.

Objectif

Cette fonction appelle en premier «changeMG()», à la page 126 la fonction de format si nécessaire et convertit les valeurs réelles et attendues. Elle vérifie si la valeur est 0, et si ce n'est pas le cas, elle renvoie "Echec" et quitte. Si les valeurs ne sont pas 0, la fonction vérifie si les valeurs sont booléennes, numériques, en Mb ou Gb, vitesse CPU en MHz (Windows uniquement) ou GHz, ou chaînes. Elle compare ensuite les valeurs et renvoie le résultat.

Syntaxe

```
succès ou échec (attendu,réel)
```

Paramètres d'entrée

Chaîne *attendue*
La valeur attendue pour la propriété de prérequis.

Chaîne *réelle*
La valeur réelle pour la propriété de prérequis.

Valeurs de retour

succès ou échec *de chaîne*
Renvoie "Succès" ou "Echec" selon que la valeur attendue est égale ou supérieure à la valeur réelle.

ppread()

Lit les contenus d'un fichier pour l'objet dictionnaire de scriptage, séparant chaque ligne dans le fichier par le séparateur spécifique du paramètre d'entrée si ce séparateur existe dans la ligne.

Objectif

Cette fonction lit chaque ligne du fichier, supprime toutes les interlignes et les espaces de fin, et vérifie si elle contient le séparateur. Si elle contient le séparateur, elle divise la ligne par le séparateur, en ajoutant chaque pièce comme un élément à l'objet dictionnaire; sinon, elle ajoute la ligne avec un élément dans l'objet dictionnaire. Elle renvoie un tableau qui contient l'objet dictionnaire comme le premier index.

Syntaxe

```
ppread (nom de fichier, séparateur)
```

Paramètres d'entrée

nom de fichier *de chaîne*
Le nom de fichier pour lire dans l'objet dictionnaire.

séparateur *de caractère*
Le caractère qui représente le séparateur permettant de diviser une ligne dans le fichier.

Valeurs de retour

tableau *mat (dictionnaire)*
Renvoie un tableau avec l'objet dictionnaire (dic) comme son premier index.

Exemple

Exemple à fournir.

readFile()

Lit chaque ligne d'un fichier dans une entrée d'index d'un tableau.

Objectif

Cette fonction ouvre le fichier et lit chaque ligne du fichier dans une entrée d'index d'un tableau. Elle renvoie le tableau.

Syntaxe

```
readFile(fileName)
```

Paramètres d'entrée

Chaîne *fileName*
Le nom du fichier à lire dans le tableau.

Valeurs de retour

Tableau *fileContents*
Renvoie le tableau avec le contenu du fichier.

unitMGTOG()

Additionne le contenu d'un tableau pour obtenir le nombre total de Mo.

Objectif

Cette fonction convertit la valeur de chaque index dans le tableau en Mo et les additionne.

Syntaxe

```
unitMGTOG(arr)
```

Paramètres d'entrée

Matrice *arr*
Contient la matrice.

Valeurs de retour

Chaîne *unitMGTOG*
Renvoie l'intégralité du contenu du tableau en Mo et ajoute `"MB"` au total.

varToString()

Crée une représentation de chaîne d'une variable. La variable à vérifier peut être une chaîne, un nombre, des objets dictionnaire de scriptage, un tableau ou un objet `CheckItem`.

Objectif

Cette fonction vérifie le type de données ou d'objet de la variable et appelle la fonction pertinente pour créer une représentation de chaîne pour ces types de données ou d'objet.

Tableau 32. Fonction appelée pour chaque type de variable.

Type de variable	Fonction appelée
Tableau	«arrayToString()», à la page 125
Objet `CheckItem`	«checkItemToString()», à la page 126
Objet dictionnaire de scriptage	«dictionaryToString()», à la page 127

Syntaxe

```
varToString(var)
```

Paramètres d'entrée

`Variable` *`var`*
Les variables prises en charge sont les suivantes : chaîne, nombre, objet dictionnaire de scriptage, tableau ou objet CheckItem.

Valeurs de retour

`Chaîne` *`vartoString`*
Renvoie une représentation de chaîne de la variable, y compris les valeurs renvoyées de toutes les fonctions appelées, le cas échéant.

Annexe F. Sous-routines des utilitaires de journalisation pour les systèmes Windows

IBM Prerequisite Scanner dispose d'un ensemble de sous-routines de journalisation dans le fichier `preq.vbs` permettant d'afficher des messages sur l'écran ou d'écrire dans le fichier journal.

tableau 33 décrit les utilitaires de journalisation.

Tableau 33. Sous-routines des utilitaires de journalisation

Sous-routine	Description	Paramètres d'entrée
`deleteLogFile`	Supprime le fichier journal s'il existe.	Aucun
`log(level, msg)`	Ecrit le message dans le fichier journal à l'aide de la fonction «fmt()», à la page 130. Le journal comprend également la date et l'heure actuelles.	• `level`, une chaîne qui définit le type de message tel qu'une information ou un avertissement • `msg`, une chaîne qui représente le message à consigner
`logDebug(msg)`	Appelle la fonction `log()`, `"DEBUG"` étant le paramètre d'entrée de niveau.	`msg`, une chaîne qui représente le message à consigner
`logError(msg)`	Appelle la fonction `log()`, `"ERROR"` étant le paramètre d'entrée de niveau.	`msg`, une chaîne qui représente le message à consigner
`logInfo(msg)`	Appelle la fonction `log()`, `"INFO"` étant le paramètre d'entrée de niveau.	`msg`, une chaîne qui représente le message à consigner
`logScreen(msg)`	Ecrit le message sur l'écran.	`msg`, une chaîne qui représente le message à écrire sur l'écran
`logScreenWith Remplacement (msg, replaceStr)`	Ecrit le message sur l'écran, le code et la chaîne du message étant les paramètres d'entrée.	• `msg`, le code du message qui représente la chaîne du message à écrire sur l'écran • `replaceStr`, la chaîne qui remplace la *%variable* de la valeur du code de message
`logScreenWith MultiReplacements (msg, replaceStrArray)`	Ecrit le message sur l'écran, le code du message et le tableau de chaînes étant les paramètres d'entrée.	• `msg`, le code du message qui représente la chaîne du message à écrire sur l'écran • `replaceStrArray`, le tableau de chaînes avec chaque index du tableau remplaçant une *%variable* de la valeur de code du message
`logWarning(msg)`	Appelle la fonction `log()`, `"WARNING"` étant le paramètre d'entrée de niveau.	`msg`, une chaîne qui représente le message à consigner

Annexe G. Utilitaire de fichiers sous-routines pour les systèmes Windows

Prerequisite Scanner a un ensemble de fichiers sous-routines communs dans le fichier `/lib/common_function.vbs` pour traiter les fichiers. Il a également un ensemble de fonctions pour traiter les fichiers.

tableau 34 décrit les utilitaires de fichiers.

Tableau 34. Utilitaire de fichiers sous-routines

Sous-routine	Description	Paramètres d'entrée
`appendToFile(text, fileName)`	Annexe le texte à la fin du fichier spécifié.	• `text`, une chaîne contenant le texte à annexer au fichier • `filename`, une chaîne représentant le nom du fichier à modifier
`writeToFile(text, fileName)`	Ecrit le texte sur le fichier spécifique en écrasant, si nécessaire, le contenu existant.	• `text`, une chaîne contenant le texte à écrire sur le fichier • `filename`, une chaîne qui représente le nom du fichier à modifier

tableau 35 indique les fonctions de fichiers qui traitent les fichiers.

Tableau 35. Les fonctions File utility

Fonction	Description
«allFiles()», à la page 124	Lit les noms de fichiers dans un répertoire spécifique à l'intérieur d'un tableau.
«filterFile()», à la page 128	Lit et filtre le contenu d'un fichier à un objet de scriptage dictionnaire.
«findNewest()», à la page 129	Recherche le plus récent fichier de configuration.
«findSuitableFile()», à la page 129	Recherche le fichier de configuration approprié pour une version produit.
«ppread()», à la page 134	Lit le contenu à un objet de scriptage dictionnaire en fractionnant davantage chaque ligne dans le fichier par le paramètre d'entrée séparateur si ce dernier existe dans la ligne.
«readFile()», à la page 135	Lit chaque ligne d'un fichier dans une entrée d'index d'une matrice.

Annexe H. Autres fonctions communes et sous-routines pour les systèmes Windows

Prerequisite Scanner possède un ensemble d'autres fonctions communes et de sous-routines qui sont utilisées dans divers fichiers.

structure les autres fonctions communes et sous-routines.

Tableau 36. Autres fonctions communes et sous-routines pour les systèmes Windows

Fonction ou sous-routine	Description
«ffirstMatch()»	Obtient la première correspondance de la chaîne de recherche dans la matrice.
«getValue()», à la page 142	Obtient l'espace disque disponible pour un répertoire spécifié.
«removeSpecialCharacters()», à la page 143	Supprime la marque ou les autres caractères spéciaux pour faciliter les comparaisons.
«versionCompare()», à la page 143	Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la propriété de prérequis réussit le contrôle prérequis. Cette fonction attend les chaînes de version séparées par des points comme paramètres d'entrée, par exemple, 1.0.0.4, 2.3, 3.40.26.7800 ou 2.3.*.

ffirstMatch()

Obtient la première correspondance de la chaîne de recherche dans la matrice.

Objectif

Cette fonction utilise une expression régulière pour rechercher le modèle, lequel est transmis comme un paramètre d'entrée dans la matrice qui est également transmise comme paramètre d'entrée. Lorsqu'elle trouve la première correspondance du modèle dans la matrice, elle ajoute la valeur issue de la matrice à l'objet dictionnaire de scriptage.

Fonctions parentes

Tableau 37. Fonctions parentes appelant ffirstMatch()

Fonction parente, Script	Description
`ud620db2level(expect, real)` dans `DB2_Version_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version DB2.
`oslevelcompare(expect, real)` dans `OS_Version_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version du système d'exploitation.

Syntaxe

```
ffirstmatch(patt,arr)
```

Paramètres d'entrée

Chaîne *patt*
Contient le modèle à rechercher.

Matrice *arr*
Contient la matrice dans laquelle rechercher le modèle de recherche.

Valeurs de retour

Dictionnaire *keys*
Retourne les clés pour l'objet dictionnaire de scriptage.

getValue()

Obtient l'espace disque disponible pour un répertoire spécifié.

Objectif

Cette sous-routine utilise l'instance de l'objet de système de fichiers pour appeler la fonction `getDriveName()` pour la paramètre d'entrée du chemin d'accès puis elle utilise la propriété `freeSpace` pour obtenir l'espace disque disponible, qui est alors converti en Mo/s. Le paramètre d'entrée de la propriété de prérequis et sa valeur sont écrits dans le fichier texte temporaire associé au fichier script.

Scripts

Tableau 38. Scripts utilisant getValue()

Script	Description
`DEZ_01040000.vbs`	Script permettant de regrouper les propriétés de prérequis et de les mettre à la disposition du fichier de configuration DEZ 01040000 uniquement
`LCM_TAD_common.vbs`	Script permettant de regrouper les propriétés de prérequis et de les mettre à la disposition des fichiers de configuration LCM 02300000 et TAD 07200000 uniquement
`TAD722_impl.vbs`	Script permettant de regrouper les propriétés de prérequis et de les mettre à la disposition du fichier de configuration TAD 07220000 uniquement

Syntaxe

```
getValue fso, sKey, drvPath
```

Paramètres d'entrée

Objet de système de fichiers *fso*
Instance de l'objet de système de fichiers.

Chaîne *sKey*
Contient une chaîne comprenant le nom de la propriété de prérequis et le symbole "égale".

Chaîne *drvPath*
Contient le chemin d'accès pour lequel obtenir l'espace disque disponible.

Valeurs de retour

Aucun

removeSpecialCharacters()

Supprime la marque ou les autres caractères spéciaux pour faciliter les comparaisons. Cette fonction se trouve dans le fichier `/lib/common.vbs`.

Objectif

Cette fonction appelle la fonction `Replace()` pour remplacer la marque, le copyright, et les symboles d'enregistrement par `""`.

Syntaxe

```
removeSpecialCharacters(s)
```

Paramètres d'entrée

Chaîne *s*
Contient la chaîne dont les caractères doivent être supprimés

Valeurs de retour

Chaîne *s*
Retourne la chaîne sans les caractères spéciaux.

versionCompare()

Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la propriété de prérequis réussit le contrôle prérequis. Cette fonction attend les chaînes de version séparées par des points comme paramètres d'entrée, par exemple, 1.0.0.4, 2.3, 3.40.26.7800 ou 2.3.*.

Objectif

Cette fonction traite d'abord les cas spéciaux où un ou les deux paramètres d'entrée sont vides et retourne des codes de retour pour représenter ces cas. Elle divise chaque version en plusieurs parties grâce au séparateur de point. Si la dernière partie de la version est le caractère générique *, la fonction considère toutes les parties manquantes de la version comme les caractères génériques, par exemple, 2.* correspond à 2.1 ou 2.3.*. Elle fait alors une boucle à travers la liste des parties pour chaque version et compare celles-ci. Puis elle retourne les codes de retour selon que la valeur attendue est inférieure, égale à, ou supérieure à la valeur réelle.

Fonctions parentes

Tableau 39. Fonctions parentes appelant la fonction versionCompare

Fonction parente, Script	Description
`cygwinVersion_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version cygwin.
`gskit7Version_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de la version 7 de gskit.

Tableau 39. Fonctions parentes appelant la fonction versionCompare (suite)

Fonction parente, Script	Description
`gkit8Version_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de la version 8 de gskit.
`internetExplorer.version_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version d'Internet Explorer.
`os.servicePack_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis du module de mise à jour du système d'exploitation.
`os.versionNumber_compare.vbs`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version du système d'exploitation.

Syntaxe

```
versionCompare(ver1,ver2)
```

Paramètres d'entrée

Chaîne *ver1*
Contient la version attendue pour une propriété de prérequis.

Chaîne *ver2*
Contient la version réelle pour une propriété de prérequis.

Valeurs de retour

Entier *0*
Retourne le code de retour 0 si les deux paramètres d'entrée sont égaux. La fonction parente retourne `"PASS"`.

Cas particulier : Retourne le code de retour 0 et ferme si les deux paramètres d'entrée sont vides.

Entier *-1*
Retourne le code de retour -1 si le premier paramètre d'entrée est inférieur au second paramètre d'entrée. La fonction parente retourne `"FAIL"`.

Cas particulier : Retourne le code de retour -1 et ferme si le premier paramètre d'entrée est vide.

Entier *1*
Retourne le code de retour 1 si le premier paramètre d'entrée est supérieur au second paramètre d'entrée. La fonction parente retourne `"PASS"`.

Cas particulier : Retourne le code de retour 1 et ferme si le second paramètre d'entrée est vide.

Annexe I. Fonctions communes pour les systèmes UNIX

Prerequisite Scanner possède un ensemble de fonctions communes dans le fichier `/lib/common_function.sh` pour l'exécution de vérifications sur des systèmes UNIX.

Tableau 40. Fonctions dans common_function.sh

Fonction	Description
«AddMG()», à la page 146	Vérifie si les paramètres d'entrée sont en Mo/s ou en Go/s et ajoute les paramètres.
«changeMG()»	Convertit le paramètre d'entrée en Mo/s ou Go/s pour les propriétés de prérequis d'espace disque ou de mémoire.
«compare()», à la page 147	Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la première valeur (réelle) est inférieure à la seconde valeur (attendue).
«cutdown()», à la page 147	Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la première valeur (réelle) est inférieure à la seconde valeur (attendue). Puis elle imprime la différence entre ces deux valeurs si la première valeur n'est pas inférieure à la seconde valeur.
«findOSInfo()», à la page 149	Recherche la version du système d'exploitation, le niveau et la version d'édition de ce dernier, ainsi que les données d'implémentation du matériel pour le système.
«mes4path()», à la page 148	Recherche l'espace disque libre pour chaque système de fichiers monté.
«mes4Path1()», à la page 148	Recherche l'espace disque libre pour chaque système de fichiers monté sur un système Solaris uniquement.
«NFScheck()», à la page 150	Vérifie l'état de système NFS des montages sur un système UNIX.
«telnetNFS()», à la page 150	Vérifie si Telnet est applicable à l'adresse IP d'un système de fichiers monté sur le port par défaut 2049.

changeMG()

Convertit le paramètre d'entrée en Mo/s ou Go/s pour les propriétés de prérequis d'espace disque ou de mémoire.

Objectif

Cette fonction vérifie d'abord que la fonction reçoit un paramètre d'entrée. Si elle reçoit un paramètre d'entrée, elle détermine si la valeur est en Mo/s ou Go/s puis convertit cette valeur en Go/s ou Mo/s respectivement.

Syntaxe

```
changeMG val
```

Paramètres d'entrée

Chaîne *$val*
Contient la valeur pour l'espace disque ou la mémoire en Mo/s ou Go/s.

Valeurs de retour

Entier *1*
Retourne 1 si la fonction ne reçoit pas de paramètre d'entrée.

Chaîne *printf "%.0fM%s",mm[1]*1024,mm[2];*
Retourne la valeur en Mo/s.

Chaîne *printf "%.2fG%s",mm[1],mm[2];*
Retourne la valeur en Go/s.

AddMG()

Vérifie si les paramètres d'entrée sont en Mo/s ou en Go/s et ajoute les paramètres.

Objectif

Cette fonction vérifie d'abord qu'elle reçoit des paramètres d'entrée. Si elle reçoit des paramètres d'entrée, elle détermine si la valeur est en Mo/s ou en Go/s puis ajoute les valeurs.

Syntaxe

```
AddMG val1 val2
```

Paramètres d'entrée

Chaîne *$val1*
Contient la valeur pour l'espace disque ou la mémoire en Mo/s ou en Go/s à ajouter à l'autre paramètre d'entrée.

Chaîne *$val2*
Contient la valeur pour l'espace disque ou la mémoire en Mo/s ou en Go/s à ajouter à l'autre paramètre d'entrée.

Valeurs de retour

Entier *1*
Retourne 1 si la fonction ne reçoit pas deux paramètres d'entrée.

Chaîne *val*
Retourne les valeurs ajoutées en Mo/s ou Go/s.

compare()

Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la première valeur (réelle) est inférieure à la seconde valeur (attendue).

Objectif

Cette fonction vérifie d'abord qu'elle reçoit deux paramètres d'entrée. Si elle les reçoit et que ces derniers ne sont pas faux tous les deux, elle détermine si les valeurs sont en Mo/s ou Go/s puis compare ces deux valeurs pour vérifier si la première valeur est inférieure à la seconde valeur. Si tel est le cas, elle retourne une valeur "false" ; sinon, elle retourne une valeur "pass".

Syntaxe

```
compare real expected
```

Paramètres d'entrée

Chaîne *réelle$*
: Contient la valeur réelle pour une propriété de prérequis.

Chaîne *attendue$*
: Contient la valeur attendue pour une propriété de prérequis.

Valeurs de retour

Entier *1*
: Retourne 1 si la fonction ne reçoit pas deux paramètres d'entrée.

Chaîne *"FAIL|PASS"*
: Retourne la chaîne `"FAIL"` si la valeur réelle est inférieure à la valeur attendue ; sinon, retourne la chaîne `"PASS"`.

cutdown()

Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare celles-ci pour déterminer si la première valeur (réelle) est inférieure à la seconde valeur (attendue). Puis elle imprime la différence entre ces deux valeurs si la première valeur n'est pas inférieure à la seconde valeur.

Objectif

Cette fonction vérifie d'abord qu'elle reçoit deux paramètres d'entrée. Si elle les reçoit, elle détermine si les valeurs sont en Mo/s ou Go/s puis les convertit en Mo/s si elles sont en Go/s. Puis elle compare ces deux valeurs pour vérifier si la première valeur est inférieure à la seconde valeur. Si tel est le cas, elle retourne une valeur "0MB" ; sinon, elle retourne la différence entre ces deux valeurs en Mo.

Syntaxe

```
cutdown real expected
```

Paramètres d'entrée

Chaîne *réelle$*
: Contient la valeur réelle pour une propriété de prérequis.

Chaîne *attendue$*
Contient la valeur attendue pour une propriété de prérequis.

Valeurs de retour

Entier *1*
Retourne 1 si la fonction ne reçoit pas deux paramètres d'entrée.

Chaîne *"FAIL|PASS"*
Retourne la chaîne `"FAIL"` si la valeur réelle est inférieure à la valeur attendue lorsqu'aucune des valeurs n'est en Mo ou Go ; sinon, retourne la chaîne `"PASS"`.

Chaîne *"OMB|Real-ExpectedMB"*
Retourne la chaîne `"OMB"` si la valeur réelle est inférieure à la valeur attendue ; sinon, retourne une représentation de chaîne de la différence entre les deux valeurs converties en Mo/s.

mes4path()

Recherche l'espace disque libre pour chaque système de fichiers monté.

Objectif

Cette fonction prend un chemin comme entrée, appelle la commande **uname** pour déterminer le système d'exploitation, puis rappelle la fonction `NFScheck` pour déterminer si le système est actif ainsi que les montages. Elle appelle ensuite la commande `df` pour déterminer l'espace disque libre pour chaque montage sur un système. Elle renvoie également la valeur pour l'espace disque libre.

Syntaxe

```
mes4Path path
```

Paramètres d'entrée

Chaîne *chemin$*
Chemin d'accès au système permettant de vérifier l'espace disque libre.

Valeurs de retour

Entier *1*
Retourne le code de retour 1 si la fonction ne reçoit pas de paramètre d'entrée.

Entier *2*
Retourne le code de retour 2 si le paramètre d'entrée n'est pas un chemin d'accès.

Chaîne *$NF*
Retourne l'espace disque libre pour chaque montage.

Chaîne *"$path Server NotAvailable Responding for $path"*
Retourne un message à l'état où le serveur pour le chemin d'accès est non disponible.

mes4Path1()

Recherche l'espace disque libre pour chaque système de fichiers monté sur un système Solaris uniquement.

Objectif

Cette fonction prend un chemin comme entrée, appelle la commande **uname** pour déterminer que le système d'exploitation est Solaris. Elle appelle ensuite la commande df pour déterminer l'espace disque libre pour chaque montage sur le système. Elle renvoie également la valeur pour l'espace disque libre.

Syntaxe

```
mes4Path1 path
```

Paramètres d'entrée

Chaîne *chemin$*
: Chemin d'accès au système permettant de vérifier l'espace disque libre.

Valeurs de retour

Entier *1*
: Retourne le code de retour 1 si la fonction ne reçoit pas de paramètre d'entrée.

Entier *2*
: Retourne le code de retour 2 si le paramètre d'entrée n'est pas un chemin d'accès.

Chaîne *$NF*
: Retourne l'espace disque libre pour chaque montage.

findOSInfo()

Recherche la version du système d'exploitation, le niveau et la version d'édition de ce dernier, ainsi que les données d'implémentation du matériel pour le système.

Objectif

Cette fonction exécute la commande `uname` et analyse sa sortie pour la version du système d'exploitation, le niveau et la version d'édition de ce dernier, ainsi que les données d'implémentation du matériel pour le système.

Syntaxe

```
findOSInfo
```

Paramètres d'entrée

Aucun

Valeurs de retour

Chaîne *$oo*
: Sortie de la commande **uname** sans les informations du système de base.

Chaîne *$kk*
: Version du système d'exploitation

Chaîne *$hh*
: Implémentation du matériel représentée par I pour le matériel i386 ou Z pour le matériel s390.

Chaîne *$rr*
: Niveau d'édition du système d'exploitation

Chaîne *$vv*
Version du niveau d'édition du système d'exploitation

telnetNFS()

Vérifie si Telnet est applicable à l'adresse IP d'un système de fichiers monté sur le port par défaut 2049.

Objectif

Cette fonction prend un protocole IP comme entrée et appelle la commande **telnet** pour tester si une connexion à distance est réussie sur le port Telnet par défaut 2049. Elle tente d'établir la connexion distance 10 fois. Si la commande **telnet** échoue, la fonction retourne une valeur `"FALSE"` ; sinon, elle renvoie la valeur `"PASS"`.

Syntaxe

```
telnetNFS ipaddr
```

Paramètres d'entrée

Chaîne *$ipaddr*
L'adresse IP permettant de vérifier si le protocole Telnet peut être effectué.

Valeurs de retour

Chaîne *"FALSE|TRUE"*
Retourne le résultat de la vérification Telnet. Elle retourne `"TRUE"` si la vérification réussit ; sinon, elle retourne `"FALSE"`.

NFScheck()

Vérifie l'état de système NFS des montages sur un système UNIX.

Objectif

Cette fonction prend un chemin comme entrée et appelle la commande de montage pour obtenir la liste des systèmes de fichiers montés. Elle appelle la commande **uname** pour déterminer le système d'exploitation. Puis elle appelle la commande **ping** pour utiliser la commande PING sur chaque système monté et si tel est le cas, elle appelle ensuite la fonction **telnetNFS** pour vérifier si une connexion distante peut être effectuée. Si l'une des actions de commande PING ou Telnet échoue, la fonction retourne une valeur `"FALSE"` ; sinon, elle retourne la valeur `"PASS"`.

Syntaxe

```
NFScheck path
```

Paramètres d'entrée

Chaîne *chemin$*
Prend un chemin d'accès valide à un répertoire comme entrée.

Valeurs de retour

Valeur booléenne *TRUE* **ou** *FALSE*
Retourne `TRUE` si la vérification de système NFS réussit, autrement dit, si elle

réussit à utiliser la commande PING sur l'adresse IP associée ou peut utiliser Telnet pour se connecter à l'adresse IP associée pour chaque système de fichiers ; sinon, elle retourne FALSE.

Exemple

Cet exemple d'utilisation provient de la fonction **mes4Path()** :

```
# vérifie s'il s'agit d'un chemin d'accès
 chemin=`echo "$1" | sed -n '/^\//p'`
 si [ -z "$path" ] ; alors
  retourne 2 ;
 ou bien
          nfs_check_status=`NFScheck $path`
          si [ "$nfs_check_status" = "TRUE" ]; alors
         cas `uname` dans
     ...
```

Annexe J. Autres fonctions pour les systèmes UNIX

Prerequisite Scanner possède un ensemble de fonctions communes dans divers fichiers.

tableau 41 structure cet ensemble de fonctions dans différents fichiers.

Tableau 41. Fonctions communes dans différents fichiers

Fonction	Description
«formatSizeDisplay()», à la page 154	Prend le paramètre d'entrée et ajoute ou réduit la partie décimale du paramètre d'entrée à deux séparateurs décimaux, par exemple, 123 Mo à 123,00 Mo ou 12,123 Mo/s à 12,12 Mo/s.
«versionCompare()», à la page 154	Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare chaque partie de la version pour déterminer si la propriété de prérequis réussit le contrôle prérequis.

tableau 42 structure l'ensemble de fonctions dans le fichier `UNIX-Linux/TAD722_impl.sh` pour l'exécution de vérifications pour Tivoli License Compliance Manager et Tivoli Asset Discovery for Distributed.

Tableau 42. Fonctions communes dans TAD722_impl.sh

Fonction	Description
«checkSunOS()», à la page 156	Vérifie si la version du système d'exploitation Solaris est destinée aux plateformes SPARC ou X86.
«checkHpux()», à la page 156	Vérifie si la version du système d'exploitation HP-UX est destinée à des plateformes IA64 ou PARISC.
«checkLinux()», à la page 156	Vérifie si la version du système d'exploitation Linux est destinée à des plateformes System p, System z, ou x86.
«getSystemId()», à la page 158	Appelle différentes fonctions de système d'exploitation pour vérifier les plateformes du système d'exploitation adéquat.
«getValue()», à la page 157	Obtient la valeur pour une clé dans un fichier spécifié si la clé existe.
«setValue()», à la page 157	Définit la valeur d'une clé dans un fichier spécifié si la propriété de prérequis existe.
«copyValue()», à la page 157	Obtient et définit la valeur pour la propriété de prérequis (clé) en fonction du produit et du système d'exploitation.
«parseDirParameter()», à la page 159	Analyse le paramètre à partir de la liste de paramètres pour l'option -p du scanner et insère sa valeur dans la liste.

Tableau 42. Fonctions communes dans TAD722_impl.sh (suite)

Fonction	Description
«getClosestExistingParentDir()», à la page 158	Obtient le répertoire parent le plus proche ou la chaîne elle-même.
«printDirSize()», à la page 159	Vérifie l'état de système NFS du système de fichiers monté puis obtient l'espace disque du système de fichiers ou son répertoire parent.

formatSizeDisplay()

Prend le paramètre d'entrée et ajoute ou réduit la taille de la partie décimale du paramètre d'entrée à deux séparateurs décimaux, par exemple, 123 Mo à 123,00 Mo ou 12,123 Mo/s à 12,12 Mo/s

Objectif

Cette fonction compte le nombre de caractères dans le paramètre d'entrée, vérifie s'il s'agit d'un nombre ou d'une chaîne, et divise la partie d'entrée en parties complète et décimale. En fonction de la partie décimale, elle étend ou réduit cette dernière en deux positions décimales. Elle renvoie également le résultat.

Scripts parents

Les scripts suivants contiennent la fonction suivante :

- `./Unix-Linux/common.sh`
- `LCM_TAD_common.sh`

Syntaxe

```
formatSizeDisplay val
```

Paramètres d'entrée

Entier *val$*
La valeur à arrondir à deux séparateurs décimaux.

Valeurs de retour

Entier *val*
Renvoie la valeur arrondie à deux séparateurs décimaux.

versionCompare()

Analyse les paramètres d'entrée qui représentent les valeurs réelle et attendue pour une propriété de prérequis et compare chaque partie d'une version pour déterminer si la première valeur (réelle) est supérieure à la seconde valeur (attendue).

Objectif

Cette fonction vérifie que la fonction reçoit deux versions en tant que paramètres d'entrée. Elle utilise awk pour analyser et diviser chaque version en différentes parties où "," est le délimiteur permettant de diviser la valeur en parties. Elle effectue ensuite une boucle pour comparer chaque partie de la première version à

la partie correspondante de la seconde version et voir si elles sont égales.

Fonctions parentes

Tableau 43. Fonctions parentes appelant la fonction versionCompare

Fonction parente, script	Description
`db2.home_compare.sh`	Compare les valeurs réelle et attendue pour l'espace disque pour la propriété de prérequis de DB2 HOME.
`oracle.Client_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis du client Oracle.
`os.locale_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis locale du système d'exploitation.
`os.MozillaVersion_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis de Mozilla Firefox.
`os.package.perl_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis du package Perl. S'appelle elle-même.
`os.RAMSize_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis de mémoire RAM.
`os.space_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis d'espace disque.
`OS_Version_compare.sh`	Compare les valeurs réelle et attendue pour la propriété de prérequis de version du système d'exploitation.

Syntaxe

```
versionCompare real expected
```

Paramètres d'entrée

Chaîne *réelle$*
Contient la valeur réelle pour une propriété de prérequis.

Chaîne *attendue$*
Contient la valeur attendue pour une propriété de prérequis.

Valeurs de retour

Entier *0*
Retourne le code de retour 0 si les valeurs réelle et attendue sont égales. La fonction parente retourne "PASS".

Cas particulier : Retourne le code de retour 0 et ferme si la fonction reçoit des paramètres d'entrée vides.

Entier *-1*
Retourne le code de retour -1 si la valeur réelle est inférieure à la valeur attendue. La fonction parente retourne `"FAIL"`.

Retourne le code de retour -1 et ferme si la fonction reçoit un second paramètre d'entrée vide.

Entier *1*
Retourne le code de retour -1 si la valeur réelle est supérieure à la valeur attendue. La fonction parente retourne `"PASS"`.

Retourne le code de retour 1 et ferme si la fonction reçoit un premier paramètre d'entrée vide.

checkHpux()

Vérifie si la version du système d'exploitation HP-UX est destinée à des plateformes IA64 ou PARISC.

Objectif

Cette fonction utilise l'option `-m` de la commande **uname** pour déterminer si le système d'exploitation HP-UX est destiné à des plateformes IA64 ou PARISC.

Syntaxe

```
checkHpux
```

Valeurs de retour

Chaîne *HPUXIA64|HPUXPARISC*
Retourne `"HPUXIA64"` si l'option `-m` est `"ia64"` ; sinon, retourne `"HPUXPARISC"`.

checkLinux()

Vérifie si la version du système d'exploitation Linux est destinée à des plateformes System p, System z, ou x86.

Objectif

Cette fonction utilise l'option `-m` de la commande **uname** pour déterminer si le système d'exploitation Linux est destiné à des plateformes System p, System z, ou x86.

Syntaxe

```
checkLinux
```

Paramètres d'entrée

Valeurs de retour

Chaîne *LINUXPSERIES|LINUXZSERIES|LINUXX86*
Retourne `"LINUXPSERIES"` si l'option `-m` est `"ppc64"` ou `"ppc"`. Elle retourne `"LINUXZSERIES"` si la valeur est `"s390x"` ou `"s390"` ; sinon, elle retourne `"LINUXX86"`.

checkSunOS()

Vérifie si la version du système d'exploitation Solaris est destinée aux plateformes SPARC ou X86.

Objectif

Cette fonction utilise l'option `-p` de la commande **uname** pour déterminer si le système d'exploitation Solaris est destiné aux plateformes SPARC ou X86.

Syntaxe

`checkSunOS`

Paramètres d'entrée

Valeurs de retour

Chaîne *SOLARISSPARC|SOLARISX86*
Retourne "SOLARISSPARC" si l'option -p est "sparc" ; sinon, retourne "SOLARISX86".

getValue()

Obtient la valeur pour une clé dans un fichier spécifié si la clé existe.

Objectif

Syntaxe

`getValue key file`

Paramètres d'entrée

Chaîne *clé$*
Contient la clé à définir.

Chaîne *fichier$*
Contient le nom du fichier qui contient la clé.

setValue()

Définit la valeur d'une clé dans un fichier spécifié si la propriété de prérequis existe.

Syntaxe

`setValue key value file`

Paramètres d'entrée

Chaîne *clé$*
Contient la propriété de prérequis à définir.

Chaîne *valeur$*
Contient la valeur de la propriété de prérequis.

Chaîne *fichier$*
Contient le nom du fichier qui comprend la propriété de prérequis.

copyValue()

Obtient et définit la valeur pour la propriété de prérequis (clé) en fonction du produit et du système d'exploitation.

Objectif

Cette fonction appelle la fonction **getValue()** afin d'obtenir la valeur pour la propriété de prérequis spécifiée pour le produit et le système d'exploitation. Elle appelle ensuite la fonction **setValue()** pour définir la valeur pour la propriété de prérequis dans le fichier Prerequisite Scanner.

Syntaxe

```
copyValue key file
```

Paramètres d'entrée

Chaîne *clé$*
Contient la clé à obtenir et définir.

Chaîne *fichier$*
Contient le nom du fichier qui contient la clé.

Valeurs de retour

getSystemId()

Appelle différentes fonctions de système d'exploitation pour vérifier les plateformes du système d'exploitation adéquat.

Objectif

Cette fonction appelle différentes fonctions du système d'exploitation pour identifier les plateformes du système d'exploitation adéquat.

Syntaxe

```
getSystemId
```

Paramètres d'entrée

Valeurs de retour

Chaîne *AIX|Linux*
Retourne `"AIX"` ou `"Linux"` si le produit est Tivoli License Compliance Manager et le système d'exploitation est soit AIX soit Linux soit "AIX" si le produit est Tivoli Asset Discovery for Distributed et le système d'exploitation est AIX.

getClosestExistingParentDir()

Obtient le répertoire parent le plus proche ou la chaîne elle-même.

Objectif

Syntaxe

```
getClosestExistingParentDir dirpath
```

Paramètres d'entrée

Chaîne *$dirpath*
Contient le chemin d'accès pour obtenir son répertoire parent ou la chaîne elle-même.

Valeurs de retour

Chaîne *dirpath*
Retourne le répertoire parent ou la chaîne elle-même

parseDirParameter()

Analyse le paramètre à partir de la liste de paramètres pour l'option -p du scanner et insère sa valeur dans la liste.

Objectif

Syntaxe

Paramètres d'entrée

Chaîne

Valeurs de retour

printDirSize()

Vérifie l'état de système NFS du système de fichiers monté puis obtient l'espace disque du système de fichiers ou son répertoire parent.

Objectif

Cette fonction appelle d'abord la fonction **NFScheck** pour déterminer l'état de système NFS du répertoire. Si l'état est "true", elle appelle la fonction **getClosestExistingParentDir** pour retourner le répertoire ou son répertoire parent, puis utilise la commande **df** pour obtenir la quantité d'espace disque libre. Elle appelle enfin la fonction **formatSizeDisplay** pour arrondir la valeur aux séparateurs décimaux.

Syntaxe

```
printDirSize dirpath
```

Paramètres d'entrée

Chaîne *$dirpath*
: Contient le chemin d'accès au répertoire pour lequel obtenir l'espace disque libre.

Valeurs de retour

Entier *dsize*
: Retourne la quantité d'espace disque libre jusqu'à deux séparateurs décimaux.

Chaîne *"NFS_NOT_AVAILABLE"*
: Retourne l'information selon laquelle le système de fichiers monté n'est pas disponible.

Annexe K. Fonctions de l'utilitaire de journalisation pour les systèmes UNIX

Prerequisite Scanner a défni un ensemble de fonctions de journalisation communes dans le fichier `/lib/common_function.sh` pour l'écriture de données de débogage et de trace dans les fichiers journaux.

tableau 44 décrit les utilitaires de journalisation.

Tableau 44. Fonctions de l'utilitaire de journalisation sur les systèmes UNIX

Fonction	Description	Paramètres d'entrée

`wrlTrace` *`log_str1 log_str2`*	Ecrit les chaînes *`log_str1`* et *`log_str2`* dans le fichier de trace, avec l'horodatage	*`log_str1`* et *`log_str2`*, chaînes de trace qui représentent l'action et le collecteur en cours d'exécution et à consigner dans le fichier de trace. Par exemple :
		`` `wrlTrace Starting os.lib` ``
		`` `wrlTrace Executing os.lib` ``
		`` `wrlDebug Starting os.lib` ``
		`` `wrlDebug Expected libXp ` ``
		`` ss=`./os.lib libXp libXp` ``
		`` `wrlTrace Finished os.lib` ``
		`echo "os.lib.libXp=$ss"`
		`` `wrlDebug Finished os.lib` ``
		`` `wrlDebug OutPutValueIs $ss` ``
		`` `wrlTrace Done os.lib` ``

`wrlTraceFuncStart` *`fn_name`*	Transmet la fonction *`fn_name`* à `wrlTrace()`	*`fn_name`*, chaîne de trace qui représente la fonction qui vient d'être appelée. Par exemple : `` `wrlTraceFuncStart "$1"` ``
`wrlTraceFuncExit` *`fn_name`*	Transmet la fonction *`fn_name`* à `wrlTrace()`	*`fn_name`*, chaîne de trace qui représente la fonction qui vient d'être terminée. Par exemple : `` `wrlTraceFuncExit "$1"` ``

`wrlDebug` *`log_str1 log_str2`*	Transmet les chaînes *`log_str1`* et *`log_str2`* à `wrlDebugGeneric()`	*`log_str1`* et *`log_str2`*, chaînes de débogage qui représentent l'action et le collecteur en cours d'exécution et à consigner dans le fichier de débogage. Par exemple :
		`` `wrlTrace Starting os.lib` ``
		`` `wrlTrace Executing os.lib` ``
		`` `wrlDebug Starting os.lib` ``
		`` `wrlDebug Expected libXp ` ``
		`` ss=`./os.lib libXp libXp` ``
		`` `wrlTrace Finished os.lib` ``
		`echo "os.lib.libXp=$ss"`
		`` `wrlDebug Finished os.lib` ``
		`` `wrlDebug OutPutValueIs $ss` ``
		`` `wrlTrace Done os.lib` ``

`wrlDebugFuncStart` *`fn_name`*	Transmet la fonction *`fn_name`* à `wrlDebug()`	*`fn_name`*, chaîne de débogage qui représente la fonction qui vient d'être appelée. Par exemple : `` `wrlDebugFuncStart "$1"` ``

Tableau 44. Fonctions de l'utilitaire de journalisation sur les systèmes UNIX (suite)

Fonction	Description	Paramètres d'entrée
`wrlDebugFuncExit` *`fn_name`*	Transmet la fonction *`fn_name`* à `wrlDebug()`	*`fn_name`*, chaîne de débogage qui représente la fonction qui vient d'être terminée. Par exemple : `` `wrlDebugFuncExit "$1"` ``
`wrlDebugFuncReturn` *`result_value`*	Ecrit le résultat *`result_value`* retourné pour la fonction dans le fichier journal	*`result_value`*, chaîne de débogage qui représente la valeur retournée depuis la fonction. Par exemple : `` `wrlDebugFuncReturn "$versionCompare"` ``
`wrlDebugFuncParam` *`param1 param2`*	Transmet les paramètres *`param1`* et *`param2`* à `wrlDebugFunc()`	*`param1`* et *`param2`*,chaînes de débogage qui représentent les titres de section analysés, les qualificatifs analysés et les arguments d'entrée analysés sur les fonctions appelées. Par exemple : `` `wrlDebugFuncParam "OSArch" "$3"` ``
`wrlDebugGeneric formatspec` *`log_str1 log_str2`*	Ecrit les chaînes *`log_str1`* et *`log_str2`* dans le fichier de débogage, mis en forme par l'argument de chaîne *`formatspec`*	• *`log_str1`* et *`log_str2`*, chaînes qui représentent les données spécifiques à consigner sur une ligne dans le fichier de débogage • *`formatspec`*, l'argument de chaîne à placer après l'horodatage mais avant l'alignement à gauche des chaînes de fichier journal et du caractère de retour à la ligne Par exemple : `` `wrlDebugGeneric "" "$1" "$2"` ``
`wrlDebugFunc` *`str`*	Transmet un caractère de tabulation et le paramètre d'entrée *`str`* à `wrlDebugGeneric()`	*`str`*, chaîne qui représente les données à consigner, autrement dit, l'état d'une vérification ou d'une action en cours de réalisation. Par exemple : `` `wrlDebugFunc "Reading config file and parsing using parse array..." ` ``
`wrlLogFuncStart` *`str`*	Transmet le paramètre d'entrée *`str`* à `wrlTraceFuncStart()` et `wrlDebugFuncStart()`	*`str`*, chaîne qui représente les données à consigner, autrement dit, le nom de la fonction en cours d'appel. Par exemple : `` `wrlLogFuncStart "main()"` ``
`wrlLogFuncExit` *`str`*	Transmet le paramètre d'entrée *`str`* à `wrlTraceFuncExit()` et `wrlDebugFuncExit()`	*`str`*, chaîne qui représente les données à consigner, autrement dit, le nom de la fonction en cours de fermeture. Par exemple : `` `wrlLogFuncExit "main()"` ``

Remarques

Ces informations concernent des produits ou des services commercialisés aux États-Unis par IBM et n'impliquent aucunement l'intention de les commercialiser dans d'autres pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service IBM puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevets couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Pour le Canada, veuillez adresser votre courrier à :

IBM Director of Commercial Relations
IBM Canada Ltd.
3600 Steeles Avenue East
Markham, Ontario
L3R 9Z7
Canada

Pour obtenir des informations sur les licences concernant les produits utilisant un jeu de caractères codé sur deux octets, contactez le service de propriété intellectuelle d'IBM de votre pays ou envoyez vos demandes par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japon

Le paragraphe suivant ne s'applique ni au Royaume-Uni ni dans aucun pays dans lequel il serait contraire aux lois locales

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFAÇON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous serait pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Ces informations contiennent des exemples de programmes d'application en langage source qui illustrent des techniques de programmation sur diverses plates-formes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plates-formes pour lesquels ils ont été écrits. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit sans redevance aucune à IBM à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation IBM.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produit et de service peuvent être des marques d'IBM ou d'autres sociétés. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web “Copyright and trademark information” à l'adresse www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript ainsi que toutes les marques incluant Adobe sont des marques d'Adobe Systems Incorporated aux Etats-Unis et/ou dans certains autres pays.

Cell Broadband Engine et Cell/B.E. sont des marques de Sony Computer Entertainment, Inc., aux Etats-Unis et/ou dans certains autres pays, et sont utilisées sous license.

Intel, le logo Intel, Intel Inside, le logo Intel Inside, Intel Centrino, le logo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium, et Pentium sont des marques d'Intel Corporation ou de ses filiales aux Etats-Unis et dans certains autres pays.

IT Infrastructure Library est une marque de The Central Computer and Telecommunications Agency qui fait désormais partie de The Office of Government Commerce.

ITIL est une marque et une marque communautaire de The Office of Government Commerce et est enregistrée U.S. Patent and Trademark Office.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Microsoft, Windows, Windows NT, et le logo Windows sont des marques de Microsoft Corporation au Etats-Unis et/ou dans certains autres pays.

UNIX est une marque de The Open Group aux Etats-Unis et dans certains autres pays.

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Informations sur l'assistance et commentaires en retour

Si un incident se produit avec votre logiciel IBM, vous souhaitez probablement le résoudre rapidement. IBM vous propose différents moyens d'obtenir l'assistance dont vous avez besoin. Par exemple, l'aide en ligne ou via IBM Support Assistant. Vous pouvez également fournir des commentaires en retour ou soumettre des demandes d'améliorations produit.

En ligne

Les sites suivants fournissent des informations d'identification et de résolution des incidents :

- Accédez à la page IBM Prerequisite Scanner à l'adresse suivante : IBM Support Portal.
- Accédez aux rubriques Prerequisite Scanner à l'adresse suivante : Service Management Connect. Vous êtes libre de consulter ces rubriques.

Accédez aux sites suivants pour effectuer des commentaires en retour, soumettre des requêtes ou échanger à propos de Prerequisite Scanner :

- Accédez aux rubriques Prerequisite Scanner à l'adresse suivante : Prerequisite Scanner at Service Management Connect Vous êtes libre de consulter ces rubriques.
- Consultez Integrated Service Management Message Board sur Service Management Connect.
- Soumettez ou révisez les demandes d'améliorations du produit pour Prerequisite Scanner à l'adresse suivante : Tivoli RFE Community.

IBM Support Assistant

IBM Support Assistant (ISA) est un plan de travail de maintenabilité logicielle local gratuit qui vous aide à résoudre les questions et les problèmes liés à l'utilisation des produits logiciels IBM. ISA permet d'accéder rapidement aux informations de support et aux outils de serviçabilité destinés à l'identification des incidents. Pour installer le logiciel ISA, accédez à l'adresse suivante : http://www.ibm.com/software/support/isa

Index

A

C

D

E

F

I

M

N

P

Q

R

S

T

V

W

X

IBM®