Prerequisite Scanner
Version 1.2

Benutzerhandbuch

Prerequisite Scanner
Version 1.2

Benutzerhandbuch

IBM

Hinweis

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die Informationen unter „Bemerkungen" auf Seite 163 gelesen werden.

Diese Ausgabe bezieht sich auf Version 1.2 von IBM Prerequisite Scanner und alle nachfolgenden Releases und Modifikationen, bis dieser Hinweis in einer Neuausgabe geändert wird.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
Perequisite Scanner Version 1.2 User's Guide,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:

TSC Germany
Kst. 2877
Juli 2012

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Kapitel 1. Übersicht über Prerequisite Scanner

IBM® Prerequisite Scanner ist ein Scanning-Tool, das Voraussetzungen für bestimmte Software ermittelt und prüft, bevor die eigentliche Implementierung stattfindet. Das Tool scannt die Hardware- und Softwarevoraussetzungen basierend auf den Werten, die für vorausgesetzte Eigenschaften definiert werden. Der Scanner zeigt die Ergebnisse in der Befehlszeilenschnittstelle an und speichert die Ergebnisse außerdem in Text- und optional in XML-Dateien. Darüber hinaus schreibt der Scanner Informationsnachrichten sowie Trace-und Debugnachrichten in Protokolldateien.

Prerequisite Scanner kann das Betriebssystem der Maschine prüfen und feststellen, ob es die richtige Version für die angegebene Software hat. Falls eine Prüfung der Voraussetzungen fehlsschlägt, schlägt der gesamte Scan fehl.

Sie können Prerequisite Scanner nach einer Installation oder zu jedem beliebigen Zeitpunkt ausführen, um Ihre aktuelle Umgebung zu überprüfen. Prerequisite Scanner erfordert nicht, dass Sie das Installationsprogramm der Software ausführen, für die Sie die Voraussetzungen überprüfen möchten.

Sie können Prerequisite Scanner erweitern, um Voraussetzungen zu prüfen, die nicht zur Kerngruppe der Prüfungen der Voraussetzungen gehören, die mit dem Scanner bereitgestellt werden.

Prerequisite Scanner ruft je nach Plattform die folgenden Typen von Scripts auf:

- Windows: VBScript und Stapelscripts
- UNIX: Shell

Anmerkung: Sie können UNIX-Scripts nicht auf Windows-Systemen ausführen, selbst wenn Sie eine UNIX-ähnliche Umgebung wie Cygwin auf den Windows-Maschinen installiert haben.

Architektur von Prerequisite Scanner

IBM Prerequisite Scanner umfasst die folgenden Hauptkomponenten: ein Script, das in einer Befehlszeilenschnittstelle ausgeführt wird, eine Gruppe von Eigenschaften für die vorausgesetzten Prüfungen, Konfigurationsdateien für vorausgesetzte Eigenschaften, vorausgesetzte Collector und vorausgesetzte Auswertungsprogramme. Die Ergebnisse der Ausführung von Prerequisite Scanner sind in verschiedenen Ausgabeformaten verfügbar.

Vorausgesetzte Eigenschaften

Vorausgesetzte Eigenschaften sind die erwarteten Werte für verschiedene Software- und Hardwarevoraussetzungen, deren Installation von den Produkten oder Lösungen vorausgesetzt wird. Beispiele für vorausgesetzte Eigenschaften sind der insgesamt auf der Maschine verfügbare Plattenspeicherplatz, die Gruppe von Ports, die auf einer Maschine nicht im Gebrauch sind und die Gruppe momentan installierter Anwendungen.

Da die Werte für diese vorausgesetzten Eigenschaften je nach Produkt verschieden sein können, werden die Eigenschaften und die zugehörigen Werte als Name/Wert-Paare mit optionalen Qualifikationsmerkmalen dargestellt. Sie sind in den

Konfigurationsdateien für vorausgesetzte Eigenschaften enthalten. In jeder Zeile steht eine vorausgesetzte Eigenschaft.

Vorausgesetzte Eigenschaften haben das folgende Format:

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

Erläuterungen:

- *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
- *property_name* ist der Name der vorausgesetzten Eigenschaft.
- *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
- *qualifier_name* ist ein optionales Attribut für die vorausgesetzte Eigenschaft. IBM Prerequisite Scanner verwendet dieses Attribut, um die vorausgesetzte Eigenschaft bzw. den Typ der für die vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren.

  Anmerkung: Sie können mehrere Qualifikationsmerkmale durch Kommas getrennt angeben. Die Gruppe der Qualifikationsmerkmale muss in eckige Klammern (`[]`) eingeschlossen werden.
- *qualifier_value* ist der Wert für das optionale Attribut. Jedes Qualifikationsmerkmal und dessen Wert muss durch einen Doppelpunkt (`:`) begrenzt werden.
- *property_value* ist der Wert für die vorausgesetzte Eigenschaft und kann eine Zeichenfolge (String) oder eine ganze Zahl (Integer) sein.

Eine vorausgesetzte Eigenschaft kann je nach Datentyp und Qualifikationsmerkmal wie folgt einen oder mehrere Werte haben:

- Einzelne ganze Zahl, z. B. `8080` für die Darstellung einer Portnummer.
- Bereich oder Gruppe ganzer Zahlen, dargestellt mithilfe von Sonderzeichen (siehe Tabelle 1)

Tabelle 1. Sonderzeichen für die Darstellung von Bereichstypen

Sonderzeichen	Beschreibung
`*`	Gibt einen Platzhalter für mehrere Werte an. `ports.*` kann beispielsweise ein Superset von Ports für ein Datenbankprodukt, `ports.DB`, und IBM WebSphere Application Server, `ports.WAS`, darstellen.
`+`	Gibt an, dass die tatsächliche Version mindestens mit dem Wert für die erwartete Version übereinstimmen muss. `os.versionNumber=5.0+` bedeutet beispielsweise, dass die Version 5.0 oder höher sein muss.
`-`	Gibt an, dass die tatsächliche Version maximal mit dem Wert für die erwartete Version übereinstimmen darf. `os.versionNumber=5.0-` bedeutet beispielsweise, dass die Version 5.0 oder früher sein muss.
`.*`	Gibt an, dass die tatsächiche Version einem der Platzhalterwerte für die erwartete Version entsprechen kann. Beispiel: `os.versionNumber=5.*, means that the version can be 5.0, 5.0.1 or 5.5.`

Einschränkung: Auf Windows-Systemen wird das Platzhalterzeichen `*` nur unterstützt, wenn es in einem regulären Ausdruck in der vorausgesetzten Eigenschaft `OS Version` verwendet wird.

- Eine Zeichenfolge, die einen der folgenden Werte für die vorausgesetzten Typen darstellen kann:
  - Numerischer Wert mit einer Einheit, z. B. `8GB` oder `10MB`
  - Anwendung, Betriebssystem, Architektur oder Paket, z. B. `IBM Lotus Symphony, RedHat Enterprise Linux 5.4, 32-bit` oder `ftp`

    Anmerkung: Eine Zeichenfolge kann auch mehrere durch Kommas getrennte Werte enthalten, z. B. eine Liste mit Anwendungen.
  - Entweder-oder-Werte, die durch eine der folgenden Kombinationen dargestellt werden, z. B. `True|False, Available|Unavailable` oder `Enabled|Disabled`

In Tabelle 2 finden Sie Beispiele für vorausgesetzte Eigenschaften.

Tabelle 2. Beispiele für vorausgesetzte Eigenschaften

Vorausgesetzte Eigenschaft	**Erläuterung**
`Disk=1GB`	Der freie Plattenspeicherplatz. Erläuterungen: • *property_name* ist `Disk` • *property_value* ist `1GB`
`user.isAdmin=True`	Gibt an, ob der angemeldete Benutzer zu einer Administratorgruppe gehört. Erläuterungen: • *prefix_identifier* ist `user` (für vorausgesetzte Benutzereigenschaften) • *property_name* ist `isAdmin` • *property_value* ist `True`

`network.availablePorts.DB=60000-60005`
`network.availablePorts.WAS=8080`
`network.availablePorts.FTP=21`
	Prüft, ob die Ports 60000-60005 für den Datenbankserver, Port 8080 für WebSphere Application Server und Port 21 für FTP verfügbar sind. Erläuterungen:
	• *prefix_identifier* ist `network` für allgemeine vorausgesetzte Eigenschaften
	• *property_name* ist `availablePorts`
	• *suffix_identifier* ist `DB` für verfügbare Datenbankports, `WAS` für den verfügbaren Port für WebSphere Application Server und FTP für den verfügbaren Port für `FTP`
	• *property_value* ist `60000-60005`, `8080` oder `21`

`os.dir.home=[dir:/home,type:permission]755+`	Prüft, ob das Ausgangsverzeichnis die Berechtigungen `drwxr-xr-x` hat. Erläuterungen:
	• *prefix_identifier* ist `os` für vorausgesetzte Eigenschaften für das Betriebssytem
	• *property_name* ist `dir`
	• *suffix_identifier* ist `home` für das zu prüfende Verzeichnis
	• *qualifier_name* ist `dir` oder
	`type (qualifizieren die vorausgesetzte Eigenschaft und den Typ der Prüfung)`
	• *qualifier_value* ist `home` oder
	`permission (Werte für die Qualifikationsmerkmale)`
	• *property_value* ist `755+`, d. h. die Zugriffsberechtigungen für das Ausgangsverzeichnis in Oktaldarstellung

Sie können vordefinierte vorausgesetzte Eigenschaften für jedes Produkt hinzufügen oder bearbeiten, für das Sie Prerequisite Scanner ausführen möchten. Sie können auch angepasste vorausgesetzte Eigenschaften erstellen und bei Bedarf Collec-

tor und Auswertungsprogramme von Prerequisite Scanner verwenden, um vorausgesetzte Eigenschaften zu scannen und zu vergleichen.

Zugehörige Konzepte:

„Vordefinierte Qualifikationsmerkmale für vorausgesetzte Eigenschaften" auf Seite 9

IBM Prerequisite Scanner stellt eine Reihe von Basisqualifikationsmerkmalen für einige vorausgesetzte Eigenschaften in einer vordefinierten Kategorie bereit. Qualifikationsmerkmale stellen Attribute der vorausgesetzten Eigenschaft dar, die Prerequisite Scanner verwendet, um die vorausgesetzte Eigenschaft oder den Typ der für diese vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren.

Vordefinierte Kategorien vorausgesetzter Eigenschaften

IBM Prerequisite Scanner stellt eine Gruppe vorausgesetzter Basiseigenschaften für verschiedene Datenkategorien bereit: allgemein, installierte Software, Betriebssystem, Benutzer, Konnektivität, Internet Explorer, Datenbankserver, Umgebungsvariablen und Netz, einschließlich plattformspezifischer Eigenschaften für Windows und UNIX.

<*prefix_identifier*> ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften.

In Tabelle 3 sind die vordefinierten Kategorien von Hardware- und Softwarevoraussetzungen beschrieben.

Tabelle 3. Basiskategorien vorausgesetzter Eigenschaften

Datenkategorie	Beschreibung	Erforderliche Präfix-ID
Allgemein	Diese Kategorie prüft allgemeine Voraussetzungen, wie z. B. die Prozessorgeschwindigkeit, den Arbeitsspeicher, den Plattenspeicherplatz und den temporären Speicherplatz. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft für die Überprüfung des Betriebssystems: `OS Version=RedHat Enterprise Linux 5.4`	Keine
Installierte Software	Diese Kategorie überprüft die Voraussetzungen für installierte Software, wie z. B. die in der Windows-Registry registrierten Programme. Außerdem wird geprüft, ob cygwin und gskit installiert sind. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft für das Durchsuchen der Betriebssystemregistry nach Programmen mit Positionen: `installedSoftware=`*`list_of_installed_programs`*	Keine
Benutzer	Diese Kategorie überprüft Voraussetzungen für Benutzer, z. B., ob der angemeldete Benutzer Administratorberechtigungen hat oder ob er der Rootbenutzer ist. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob der angemeldete Benutzer zur Administratorgruppe gehört: `user.isAdmin=True`	`user`
Betriebssystem	Diese Kategorie überprüft Voraussetzungen für das Betriebssystem, z. B. Version, Architektur, Gesamtspeicher, verfügbaren Hauptspeicher und physischen Gesamthauptspeicher. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob der ferne Registry-Service aktiv ist: `os.isServiceRunning.remoteRegistry=True`	`os`
Konnektivität	Diese Kategorie überprüft die Voraussetzungen für die Konnektivität, z. B., ob Telnet aktiv ist und zu welchen IP-Adressen und Ports der Scanner eine Verbindung herstellen kann.	Keine

Tabelle 3. Basiskategorien vorausgesetzter Eigenschaften (Forts.)

Datenkategorie	Beschreibung	Erforderliche Präfix-ID
Netz	Diese Kategorie überprüft die allgemeinen Netzvoraussetzungen für alle Plattformen, z. B., ob Ports verfügbar sind. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob der Port `8080` für IBM WebSphere Application Server verfügbar ist: `network.availablePorts.was=8080`	`network`
Windows-Netz	Diese Kategorie überprüft Voraussetzungen für das Windows-Netz, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob in mindestens einem Adapter mit einer gültigen IP-Adresse NetBIOS als Protokoll aktiviert ist: `network.netBIOSEnabled=True`	`network`
UNIX-Netz	Diese Kategorie überprüft Voraussetzungen für das UNIX-Netz, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob der lokale Host auf das Ping-Protokoll reagiert: `network.pingLocalhost=True`	`network`
Internet Explorer	Diese Kategorie überprüft die Voraussetzungen für Microsoft Internet Explorer, wie z. B. die Version. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob Internet Explorer Version `7.0` ist: `internetExplorer.version=7.0`	`internetExplorer`
Datenbankserver, DB2	Diese Kategorie überprüft die Voraussetzungen für DB2, z. B. die Version. Das folgende Beispiel zeigt vorausgesetzte Eigenschaft, mit der geprüft wird, ob die DB2-Version mindestens `9.5` ist: `DB2 Version=9.5.*`	`DB2`
Datenbankserver, Oracle	Diese Kategorie überprüft die Voraussetzungen für Oracle, z. B. die Version. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob die Oracle-Clientversion mindestens `9.2.0.8` ist: `oracle.Client=9.2.0.8+`	`Oracle`
Umgebungsvariablen	Diese Kategorie überprüft Voraussetzungen für Umgebungsvariablen, z. B., ob die Umgebungsvariable gesetzt ist. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob der Klassenpfad die Derby-JAR-Datei enthält: `env.classpath.derbyJAR=False`	`env`
Autonomic Deployment Engine	Diese Kategorie überprüft die Voraussetzungen für Autonomic Deployment Engine, z. B., ob Autonomic Deployment Engine installiert ist, oder die Installationseinheit für Tivoli Integrated Portal. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob die Installationseinheit für Tivoli Integrated Portal Version 2.1.1.0 oder 2.1.1.1 auf einem Windows-System installiert ist: `de.installationUnit=regex{.*C37109911C8A11D98E1700061BDE7AEA.*|.*TIP 2.1.1.0.*|.*TIP 2.1.1.1.*}`	`de`
Datenbankserver, MS SQL	Diese Kategorie überprüft die Voraussetzungen für MS SQL, wie z. B. die Version. Das folgende Beispiel zeigt die vorausgesetzte Eigenschaft, mit der geprüft wird, ob MS SQL Server die Version SQL Server 2008 R2 Developer Edition hat: `mssql.Server=10.50.1600.1`	`mssql`

Vordefinierte Subtypen für vorausgesetzte Eigenschaften

IBM Prerequisite Scanner stellt eine Reihe von Basissubtypen für einige vorausgesetzte Eigenschaften in einer vordefinierten Kategorie bereit. Subtypen kategorisieren vorausgesetzte Eigenschaft näher, z. B. Kategorisierung nach den Subtypen Anwendung, Dienstprogramm oder Service.

Angenommen, Sie haben eine vorausgesetzte Eigenschaft für verfügbare Netzports. Sie können weiter kategorisieren, dass die vorausgesetzte Eigenschaft verfügbare Ports für einen Datenbankserver, einen Anwendungsserver oder ein Protokoll sucht.

<*suffix_identifier*> ist eine optionale ID für einen Subtyp im Namen der vorausgesetzten Eigenschaft.

In Tabelle 4 sind die vordefinierten Subtypen für verschiedene Kategorien vorausgesetzter Eigenschaften beschrieben, einschließlich <*suffix_identifier*>.

Tabelle 4. Vordefinierte Subtypen

Subtyp der vorausgesetzten Eigenschaft	**Suffix-ID**	**Plattform**	**Beschreibung**	**Gültige Werte für den Subtyp**
Kategorie Plattformunabhängiges Netz				
`network.availablePorts.`*`Anwendungstyp`*	*app_type*	Alle	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Port bzw. Portbereich nicht überwacht wird oder ob er für den Anwendungstyp *app_type* verfügbar ist.	Zeichenfolge für die Darstellung von *app_type*, z. B.: • `DB2` überprüft die Ports für den DB2-Datenbankserver • `WAS` überprüft die Ports für WebSphere Application Server • `ftp` überprüft den FTP-Port
`network.portsInUse.`*`app_type`*	*app_type*	Alle	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Port bzw. Portbereich überwacht wird oder ob er für den Anwendungstyp *Anwendungstyp* im Gebrauch ist.	Zeichenfolge für die Darstellung von *app_type*, z. B.: • `DB2` überprüft die Ports für den DB2-Datenbankserver • `WAS` überprüft die Ports für WebSphere Application Server • `ftp` überprüft den FTP-Port
Kategorie Betriebssystem				
`os.dir.`*`dir_name`*	*dir_name*	UNIX	Verwenden Sie diese Namenskonvention für die Überprüfung des Dateisystems *dir_name*. Der Wert für die vorausgesetzte Eigenschaft verwendet vordefinierte Qualifikationsmerkmale.	Zeichenfolge für die Darstellung von *dir_name*, z. B.: • `tmp` • `home`
`os.file.`*`script_name`*	*script_name*	UNIX	Verwenden Sie diese Namenskonvention, um zu prüfen, ob das Script *script_name* auf der Maschine verfügbar ist.	Zeichenfolge für die Darstellung von *script_name*, z. B.: • `bash` • `expect` • `gzip` • `tar`

Tabelle 4. Vordefinierte Subtypen (Forts.)

Subtyp der vorausgesetzten Eigenschaft	Suffix-ID	Plattform	Beschreibung	Gültige Werte für den Subtyp
`os.isServiceRunning.`*`service_name`*	*service_name*	Windows	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Service *service_name* auf der Maschine aktiv ist.	Zeichenfolge für die Darstellung von *service_name*, z. B.: • `remoteRegistry` • `DNSClient` • `terminalServices`
`os.lib.`*`lib_name_version`*	*lib_name _version*	UNIX	Verwenden Sie diese Namenskonvention, um zu prüfen. ob die unterstützte Version der Bibliothek *lib_name_version* auf der Maschine installiert ist.	Zeichenfolge für die Darstellung von *lib_name_version*, z. B. in Fettschrift: • 32-Bit-Bibliothek **`libstdc++.so.#`** • 64-Bit-Bibliothek **`libstdc++.so.#`** • 32-Bit-Bibliothek **`libXft.so.#`** • 32-Bit-Bibliothek **`libXtst.so.#`** • 64-Bit-Bibliothek **`libaio.so.#`** • 32-Bit-XLC-Laufzeitversion **`xlC.rte`** • 32-Bit-Laufzeitumgebung **`xlC.aix50.rte`** für AIX Version 5.3 • 32-Bit-Laufzeitumgebung **`xlC.aix61.rte`** für AIX Version 6.1 • AIX-IOCP-Bibliothek **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, die ISO-Codedateigruppe für das AIX-Basisbetriebssystem `regex {`*`str`*`}`, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für den Bibliotheksnamen darstellt, z. B.: `regex {.*libgcc.*}` Prüft, ob eine Version der GCC-Low-Level-Laufzeitbibliothek `libgcc` für dieses Betriebssystem existiert.

Tabelle 4. Vordefinierte Subtypen (Forts.)

Subtyp der vorausgesetzten Eigenschaft	Suffix-ID	Plattform	Beschreibung	Gültige Werte für den Subtyp
`os.package.` *`package_name`*	*package_name*	UNIX	Verwenden Sie diese Namenskonvention, um zu prüfen. ob die unterstützte Version des Pakets *package_name* auf der Maschine installiert ist.	Zeichenfolge für die Darstellung von *package_name*, z. B. in Fettschrift: • **`bash`** für die Shell • **`expect`** für das TCL-Erweiterungspaket • **`libgcc`** für das GCC-Low-Level-Laufzeitpaket • **`openssh`** für die Open-Source-Secure-Shell • **`openssl`** für das Open-Source-Toolkit für SSL/TLS • **`perl`** für das Perl-Scripting-Paket • **`rpm`** für RPM- oder RPM-Build-Pakete • **`telnet`** für das Telnet-Paket • **`wget`** für das GNU-Paket für Dateiabruf
`os.space.` *`dir_name`*	*dir_name*	UNIX	Verwenden Sie diese Namenskonvention, um den verfügbaren Plattenspeicherplatz für das angegebene Dateisystem *dir_name* zu überprüfen. Der Wert für die vorausgesetzte Eigenschaft verwendet vordefinierte Qualifikationsmerkmale.	Zeichenfolge für die Darstellung von *dir_name*, z. B.: • `usr` • `home` • `tmp` • `var`

Vordefinierte Qualifikationsmerkmale für vorausgesetzte Eigenschaften

IBM Prerequisite Scanner stellt eine Reihe von Basisqualifikationsmerkmalen für einige vorausgesetzte Eigenschaften in einer vordefinierten Kategorie bereit. Qualifikationsmerkmale stellen Attribute der vorausgesetzten Eigenschaft dar, die Prerequisite Scanner verwendet, um die vorausgesetzte Eigenschaft oder den Typ der für diese vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren.

Sie können beispielsweise eine vorausgesetzte Eigenschaft für ein Dateisystem haben. Sie können basierend auf dem Dateisystemnamen und den Zugriffsberechtigungsattributen qualifizieren, welche Prüfung für diese vorausgesetzte Eigenschaft durchgeführt werden soll. Außerdem können Sie basierend auf den Attributen für den Dateisystempfad und die Einheit qualifizieren, welcher Typ von Maßeinheit für die Überprüfung des verfügbaren Plattenspeicherplatzes zu verwenden ist.

Qualifikationsmerkmale unterstützen die Anpassung an die Anforderungen Ihrer Umgebung und verhindern, dass der Scanner implizite Annahmen für die Attribute mehrdimensionaler Voraussetzungen trifft, wie z. B. den Standardpfad und die Zugriffsberechtigungen. Sie können die Werte für die vordefinierten Qualifikations-

merkmale ändern, aber Sie können keine neuen Qualifikationsmerkmale zum vorhandenen Satz vordefinierter Qualifikationsmerkmale für eine vorausgesetzte Eigenschaft hinzuzufügen.

Qualifikationsmerkmale müssen das folgende Format haben:

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

Erläuterungen:

- *qualifier_name* ist ein optionales Attribut für die vorausgesetzte Eigenschaft, die IBM Prerequisite Scanner verwendet, um die vorausgesetzte Eigenschaft oder den Typ der für die vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren.
- *qualifier_value* ist der Wert für das optionale Attribut.

  Der Wert für das Qualifikationsmerkmal kann auch ein Name/Wert-Paar sein, um je nach Benutzertyp mehrere gültige Werte zu unterstützen. Abhängig davon, ob der Benutzer ein Rootbenutzer oder ein Benutzer ohne Rootberechtigung ist, können beispielsweise verschiedene Pfade für das Ausgangsverzeichnis angegeben werden.
- *property_value* ist der Wert für die vorausgesetzte Eigenschaft und kann eine Zeichenfolge (String) oder eine ganze Zahl (Integer) sein.

Jedes Qualifikationsmerkmal mit seinem Wert muss durch einen Doppelpunkt (`:`) begrenzt werden. Sie können mehrere Qualifikationsmerkmale durch Kommas getrennt angeben. Die Gruppe der Qualifikationsmerkmale muss in eckige Klammern (`[]`) eingeschlossen werden.

In Tabelle 5 sind die vordefinierten Qualifikationsmerkmale für verschiedene Kategorien vorausgesetzter Eigenschaften beschrieben. Einige vorausgesetzte Eigenschaften verwenden vordefinierte Typen auch, um eine vorausgesetzte Eigenschaft näher zu kategorisieren.

Wichtig: Sie können die vordefinierten Qualifikationsmerkmale nicht mit anderen vordefinierten vorausgesetzten Eigenschaften verwenden.

Tabelle 5. Vordefinierte Qualifikationsmerkmale

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Qualifikationsmerkmale und Werte
Kategorie Betriebssystem mit vordefiniertem Subtyp			

`os.dir.dir_name`	UNIX	Überprüft das Dateisystem *dir_name* basierend auf den folgenden Qualifikationsattributen:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystems
		• Attribut `type` zum Bestimmen des zu prüfenden Dateisystemattributs, z. B. `<octal_digits>` für die Oktaldarstellung der Zugriffsberechtigungen für dieses Dateisystem
		<*dir_name*> kann beispielsweise Folgendes darstellen:
		• `tmp`
		• `home`
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[dir:dir_name, type:permission] octal_digits+`
			
			Verwenden Sie beispielsweise die folgende Zeichenfolge, um zu prüfen, ob das Ausgangsverzeichnis die Berechtigungen `drwxr-xr-x` hat:
			`os.dir.home=[dir:/home, type:permission]755+`

Tabelle 5. Vordefinierte Qualifikationsmerkmale (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Qualifikationsmerkmale und Werte

`os.space.`
`dir_name`
	UNIX	Überprüft den verfügbaren Plattenspeicherplatz für das angegebene Dateisystem *Verzeichnisname* basierend auf einem oder mehreren der folgenden Qualifikationsattribute:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystempfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
		
		Der Wert für das Attribut `dir` hängt vom angemeldeten Benutzer ab. Deshalb ist der Wert ein Name/Wert-Paar für die Darstellung des Benutzertyps, z. B. Rootbenutzer oder Benutzer ohne Rootberechtigungen, und des zugehörigen Pfads.
		
		dir_name kann beispielsweise Folgendes darstellen:
		• `usr`
		• `home`
		• `tmp`
		• `var`
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Rootbenutzers:
			`[dir:`**`root`**`=`*`dir_path`*`,`
			`unit:`*`unit_name`*`]`
			`disk_space`
			
			Beispiel:
			`os.space.usr=`
			`[dir:root=/usr/ibm/common/acsi,`
			`unit:GB]200`
			
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Benutzers ohne Rootberechtigung:
			`[dir:`**`non_root`**`=`*`dir_path`*`,`
			`unit:`*`unit_name`*`]`
			`disk_space`
			
			Beispiel:
			`os.space.home=`
			`[dir:non_root=USERHOME/.acsi_HOST,`
			`unit:MB]200`
			
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat mit nur einem einzigen Qualifikationsmerkmal:
			`[dir:`*`dir_path`*`]`
			`disk_space` `MB`
			
			Beispiel:
			`os.space.home=[dir:/home/sat]250MB`

Kategorie Betriebssystem ohne vordefiniertem Subtyp			

`os.mountcheck`	UNIX	Prüft basierend auf den folgenden Qualifikationsattributen, ob das Dateisystem angehängt ist:
		• Attribut `drive` zum Bestimmen des Verzeichnisses, an das das Dateisystem angehängt ist
		• Attribut `nosuid` zum Bestimmen, ob die Mountoption beim Anhängen des Dateisystems gesetzt ist
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[drive:`*`dir_name`*`,`
			`mount_option``:`
			`false|true]`
			`True|False`
			
			Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Verzeichnis `/home` angehängt ist und die Option `nosuid` nicht gesetzt ist:
			`os.mountcheck=[drive:/home,`
			`nosuid:false]True`

Tabelle 5. Vordefinierte Qualifikationsmerkmale (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Qualifikationsmerkmale und Werte

`os.SELinux`	Linux	Überprüft den Aktivierungsstatus des Linux-Features Security-Enhancement basierend auf den folgenden Qualifikationsattributen:
		• Attribut `source` zum Bestimmen des für das relevante Betriebssystem zu verwendenden Befehls
			• Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[source:Command] Disabled|Enabled`
			Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Feature inaktiviert ist oder einen Berechtigungsstatus unter dem Betriebssystem Red Hat oder SUSE hat:
			`os.SELinux=[source: Command]Disabled`
			• Zeichenfolge ohne Qualifikationsmerkmal, in der das Betriebssystem eine generische Linux-Variante ist:
			`os.SELinux=Disabled`

`os.ulimit`	UNIX	Verwenden Sie diese Namenskonvention, um auf der Basis der folgenden Qualifikationsattribute zu prüfen, ob eine unbegrenzte Anzahl an Prozessen ausgeführt werden kann:
		• Attribut `type` zur Bestimmung des zusätzlich zu prüfenden Grenzwerts. `filedescriptorlimit` überprüft beispielsweise den Grenzwert für die Anzahl der Dateideskriptoren, die Prozesse öffnen können.
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[type:`*`limit_name`*`]`*`limit_value`*`, limited|unlimited`
			Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob der Grenzwert für Dateideskriptoren höher als 8192 mit einer unbegrenzten Prozessanzahl ist:
			`os.ulimit= [type:filedescriptorlimit] 8192+,unlimited`
			Im Folgenden sind die gültige Typen für die Grenzwertüberprüfung aufgelistet, wobei *limit_name* den Typ des Grenzwerts darstellt:
			• `ALL` (überprüft alle Grenzwerte)
			• `corefilesizelimit`
			• `datasegmentlimit`
			• `filedescriptorlimit`
			• `filesizelimit`
			• `hardlimit`
			• `processlimit`
			• `maxmemorysizelimit`
			• `maxprocesseslimit`
			• `stacksizelimit`
			• `threadlimit`

Kategorie Allgemein ohne vordefinierten Subtyp			

Tabelle 5. Vordefinierte Qualifikationsmerkmale (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Qualifikationsmerkmale und Werte

`Disk`	Windows	Der freie Plattenspeicherplatz mit den folgenden optionalen Qualifikationsattributen.
		• Attribut `dir` zum Bestimmen des zu prüfenden Verzeichnispfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[dir:`*`dir_path`*`, unit:`*`unit_name`*`]`
			`disk_space`
			Beispiel:
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			Numerisches Format in MB oder GB:
			`<disk_space>MB|GB`
			Beispiel:
			`Disk=250MB`

Produktcodes

IBM Prerequisite Scanner verwendet Mehrzeichencodes in Dateinamen und Parameternamen, um Produkte und Komponenten zu identifizieren und festzulegen, welcher Typ von Konfigurationsdatei zu verwenden ist.

product_code

Dies ist die Variable für die Darstellung eines Produktcodes auf Windows- oder UNIX-Systemen. Produktcodes identifizieren das Produkt, eine individuelle Plattform wie Windows, AIX, HP-UX, Linux oder Solaris und optional die Version des Betriebssystems, das von diesem Produkt unterstützt wird. Sie werden in der Datei `codename.cfg` gespeichert. Jedes Produkt, das mehrere Plattformen unterstützt, hat mehrere Produktcodes, die jeweils ein Produkt, eine Plattform und eine Version des Betriebssystems identifizieren.

Die Produktcodes `COD`, `COK` und `COX` identifizieren beispielsweise unterstützte Betriebssysteme und Versionen für 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
```

Wenn Sie Prerequisite Scanner ausführen, übergeben Sie den Produktcode und optional die Produktversion als Eingabeparameter. Der Scanner prüft, ob der Produktcode in der Datei `codename.cfg` vorhanden ist. Wenn er den Code nicht findet, wird der Scanner auf UNIX-Systemen beendet. Auf Windows-Systemen wird der Scanner in diesem Fall nicht beendet.

Der Scanner verwendet dann die Eingabeparameter, um die Konfigurationsdatei im Verzeichnis *ips_root*`/Windows|UNIX_Linux` zu suchen. Der Dateiname enthält den Produktcode und die Produktversion, die als Eingabeparameter angegeben wurden. Wenn Sie den optionalen Parameter für die Produktversion nicht übergeben, verwendet der Scanner die neueste Version der Konfigurationsdatei, die er in diesem Verzeichnis findet. Prerequisite Scanner beginnt dann mit dem Scan.

Anmerkung: Nur auf Windows-Systemen: Wenn der Produktcode nicht in der Datei `codename.cfg` vorhanden ist, aber eine Konfigurationsdatei mit

dem Produktcode im Namen existiert, zeigt Prerequisite Scanner den Produktcode und die Versionsnummer mit "not defined" für den Produktnamen in der Ausgabe an.

Konfigurationsdateien von Prerequisite Scanner

Die Konfigurationsdateien von IBM Prerequisite Scanner für einzelne Plattformen enthalten die vorausgesetzten Eigenschaften und deren erwartete Werte für jede Plattform, die von dem Produkt unterstützt wird. Prerequisite Scanner stellt eine vordefinierte Gruppe von Konfigurationsdateien bereit, die Sie bearbeiten können. Sie müssen die Konfigurationsdateien für neue Produkte und Plattformen, die unterstützt werden sollen, erstellen.

Konfigurationsdateien haben die Dateierweiterung `.cfg`. Sie speichern sie im Verzeichnis *ips_root*`/`*<OS>*, wobei *<OS>* der Name des Betriebssystems ist, z. B. `Windows` oder `UNIX_Linux`.

Konfigurationsdateien müssen den folgenden Regeln entsprechen:

- Die Dateierweiterung muss `.cfg` sein.
- Namenskonventionen für den Dateinamen:

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

  Erläuterungen:
  - *product_code*

    Dies ist die Variable für die Darstellung eines Produktcodes auf Windows- oder UNIX-Systemen. Produktcodes identifizieren das Produkt, eine individuelle Plattform wie Windows, AIX, HP-UX, Linux oder Solaris und optional die Version des Betriebssystems, das von diesem Produkt unterstützt wird. Sie werden in der Datei `codename.cfg` gespeichert. Jedes Produkt, das mehrere Plattformen unterstützt, hat mehrere Produktcodes, die jeweils ein Produkt, eine Plattform und eine Version des Betriebssystems identifizieren.
  - *<version>* ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100.
- Die vorausgesetzten Eigenschaften müssen in Abschnitten gruppiert werden, die der Namenskonvention für Abschnittstitel entsprechen müssen.
- Das Standardformat für jede vorausgesetzte Eigenschaft ist ein Name/Wert-Paar mit optionalen Qualifikationsmerkmalen und jeweils einer Eigenschaft in jeder Zeile:

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

Beispiel für eine Konfigurationsdatei ohne Abschnitte

In diesem Beispiel werden vorausgesetzte Eigenschaften überprüft, aber es wird nicht zwischen verschiedenen vorausgesetzten Eigenschaften für die erforderlichen Betriebssystemversionen unterschieden:

```
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
```

Zugehörige Konzepte:

„Abschnitte in Konfigurationsdateien"
Vorausgesetzte Eigenschaften können in einer Gruppe von Abschnitten in Konfigurationsdateien gruppiert werden, wobei jeder Abschnitt für eine Datentypkategorie steht. Abschnitte in Konfigurationsdateien sind optional.

Abschnitte in Konfigurationsdateien

Vorausgesetzte Eigenschaften können in einer Gruppe von Abschnitten in Konfigurationsdateien gruppiert werden, wobei jeder Abschnitt für eine Datentypkategorie steht. Abschnitte in Konfigurationsdateien sind optional.

Die Namenskonvention für die Abschnittstitel ist wie folgt:

[*category_name*:*category_value*]

Erläuterungen:

- *category_name* ist der Mehrzeichencode, der die Datentypkategorie darstellt.
- *category_value* ist der Mehrzeichencode, der einen zulässigen Wert für die Kategorie darstellt.

  Anmerkung: In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

Die einzelnen Kategorienamen und ihre Werte müssen mit dem folgenden Zeichen voneinander getrennt werden:

```
: Doppelpunkt und eingeschlossen in [] eckige Klammern
```

Sie können mehrere Datentypkategorien verwenden, indem Sie Abschnittstitel kombinieren. Auf diese Weise werden die vorausgesetzten Eigenschaften auf die angegebenen Kategorien beschränkt.

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

Angenommen, Sie möchten vorausgesetzte Eigenschaften angeben, die für eine Maschine gelten, auf der das Betriebssystem SUSE Linux Enterprise Server Version 11, Itanium, 32 Bit ausgeführt wird:

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

Für alle Plattformen können Sie das logisches Oder-Zeichen | verwenden, um eine oder beide Datentypkategorien zu verwenden. Wenn Sie beispielsweise Umgebungsvariablen auf `True` gesetzt haben, ist die Kombination der Abschnittstitel wie folgt:

- **UNIX-Systeme**

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

- **Windows-Systeme**

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

Wichtig: Die Position des logischen Oder-Zeichens | ist auf Windows- und UNIX-Systemen jeweils anders. Auf UNIX-Systemen wird die gesamte Gruppe von Abschnittstiteln in eckige Klammern [] eingeschlossen, und die Abschnittstitel wer-

den durch das Symbol getrennt. Auf Windows-Systemen begrenzt das Symbol die vollständigen Abschnittstitel, die jeweils in eckige Klammern [] eingeschlossen sind.

Nur auf Windows-Systemen können Sie das logische Nicht-Zeichen ! verwenden, um eine Datentypkategorie auszuschließen. Wenn Sie beispielsweise die Variante Windows Server 2003 R2 ausschließen möchten, ist die Kombination der Abschnittstitel wie folgt: `[OSType:Windows Server 2003][!OSType:Windows Server 2003 R2]`

In Tabelle 6 auf Seite 17 sind die unterstützten Datentypkategorien und die zugehörigen zulässigen Werte beschrieben.

Tabelle 6. Unterstützte Datentypkategorien und Werte

Datentypkategorie	Beschreibung	Zulässige Werte

`OSType`	Der Betriebssystemtyp.	• UNIX
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle UNIX-Plattformen gelten, einschließlich AIX, HP-UX, Linux und Solaris.
		`[OSType:UNIX]`
		• AIX
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle AIX-Betriebssystemvarianten gelten, z. B.:
		`[OSType:AIX]`
		• HP-UX
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle HP-UX-Betriebssystemvarianten gelten, z. B.:
		`[OSType:HP-UX]`
		• LINUX
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Linux-Betriebssystemvarianten gelten, z. B.:
		`[OSType:LINUX]`
		• RedHat
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems RedHat Linux gelten, z. B.:
		`[OSType:RedHat]`
		• RedHatEnterpriseLinuxServer
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems RedHat Enterprise Linux Server gelten, z. B.:
		`[OSType:RedHatEnterpriseLinuxServer]`
		• SUSE
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems SUSE Linux gelten, z. B.:
		`[OSType:SUSE]`
		• SUSELinuxEnterpriseServer
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems SUSE Linux Enterprise Server gelten, z. B.:
		`[OSType:SUSELinuxEnterpriseServer]`
		• Solaris
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Solaris-Betriebssystemvarianten gelten, z. B.:
		`[OSType:Solaris]`

Tabelle 6. Unterstützte Datentypkategorien und Werte (Forts.)

Datentypkategorie	Beschreibung	Zulässige Werte

		• Windows
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Windows-Betriebssysteme gelten, z. B.:
		`[OSType:Windows]`
		• Windows 2000 Workstation (Version 5.0.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows 2000 gelten, z. B.:
		`[OSType:Windows 2000]`
		• Windows XP Workstation (Version 5.1.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows XP Professional (32 Bit) gelten, z. B.:
		`[OSType:Windows XP]`
		• Windows XP Workstation (Version 5.2.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows XP Professional (64 Bit) gelten, z. B.:
		`[OSType:Windows XP]`
		• Windows Vista Workstation (Version 6.0.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows Vista gelten, z. B.:
		`[OSType:Windows Vista]`
		• Windows 7 Workstation (Version 6.1.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows 7 gelten, z. B.:
		`[OSType:Windows 7]`
		• Windows 2000 Server (Version 5.0.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows 2000 Server gelten, z. B.:
		`[OSType:Windows 2000]`
		• Windows Server 2003 (Version 5.2.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows Server 2003 gelten, z. B.:
		`[OSType:Windows Server 2003]`
		• Windows Server 2003 R2 (Version 5.2.* und andere Typen mit der Betriebssystembeschreibung R2)
		Gibt an, dass alle Eigenschaften in dieser Kategorie nur für die Variante Windows Server 2003 des Betriebssystems gelten, z. B.:
		`[OSType:Windows Server 2003 R2]`

Tabelle 6. Unterstützte Datentypkategorien und Werte (Forts.)

Datentypkategorie	Beschreibung	Zulässige Werte

		• Windows Server 2008 (Version 6.0.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie für alle Varianten des Betriebssystems Windows Server 2008 gelten, z. B.:
		`[OSType:Windows Server 2008]`
		• Windows Server 2008 R2 (Version 6.1.*)
		Gibt an, dass alle Eigenschaften in dieser Kategorie nur für die Variante Windows Server 2008 des Betriebssystems gelten, z. B.:
		`[OSType:Windows Server 2008 R2]`
		• *<OS_Name_Version>*
		Gibt an, dass alle Eigenschaften in dieser Kategorie für die angegebene Version des Betriebssystems gelten, z. B.:
		`[OSType:RedHatEnterpriseLinuxServer4.2]`
		Anmerkung: Das Sonderplatzhalterzeichen * ist zulässig, um mehrere Versionen anzugeben.

`OSArch`	Die Architektur für das Betriebssystem.	• 32 Bit, z. B.:
		`[OSArch:32-bit]`
		• 64b Bit, z. B.:
		`[OSArch:64-bit]`

`CPU`	Der generische Prozessorfamilienname.	Itanium, z. B.: `[CPU:Itanium]`
`CPUArch`	Die Architektur für den Prozessor.	Architektur für 64-Bit-PowerPC- und -Power-Architecture-Prozessoren, d. h.: • ppc4 • POWER4 • POWER5 • POWER6 • POWER7 Beispiel: `[CPUArch:ppc4]`
`@<EnvVar_Name>`	Die Umgebungsvariable für ein Produkt.	Muss den Regeln für dieses Produkt entsprechen, z. B.: `[@TPAE_DB_SERVER:True]`

Beispielkonfigurationsdatei für Windows mit Abschnitten

In diesem Beispiel werden Abschnitte verwendet, um vorausgesetzte Eigenschaften für Windows-Maschinen und dann für Maschinen zu kategorisieren, auf denen bestimmte Versionen von Windows ausgeführt werden.

```
#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+
```

Wenn Sie Prerequisite Scanner ausführen, scannt und überprüft der Scanner verschiedene vorausgesetzte Eigenschaften abhängig vom Betriebssystem und von der Version, die auf der Maschine installiert ist.

In Tabelle 7 sind die verschiedenen Abschnitte beschrieben, die die vorausgesetzten Eigenschaften enthalten, die basierend auf dem Beispiel geprüft werden.

Tabelle 7. Gescannte Abschnitte einer Konfigurationsdatei für Windows

Plattform oder Betriebssystem	Abschnitte mit vorausgesetzten Eigenschaften
Maschine mit Windows XP und höher	`[OSType:Windows]`
Maschine mit Windows Vista ausschließlich	`[OSType:Windows]` `[OSType:Windows Vista]`

Beispielkonfigurationsdatei für UNIX mit Abschnitten

Dieses Beispiel enthält vorausgesetzte Eigenschaften für alle Plattformen, einzelne Plattformen und Versionen von Betriebssystemen für ein bestimmtes Produkt.

```
# 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+
```

Wenn Sie Prerequisite Scanner ausführen, scannt und überprüft der Scanner verschiedene vorausgesetzte Eigenschaften abhängig vom Betriebssystem und von der Version, die auf der Maschine installiert ist.

In Tabelle 7 auf Seite 20 sind die verschiedenen Abschnitte beschrieben, die die vorausgesetzten Eigenschaften enthalten, die basierend auf dem Beispiel geprüft werden.

Tabelle 8. Gescannte Abschnitte einer Konfigurationsdatei für UNIX

Betriebssysteme und Versionen	Abschnitte mit vorausgesetzten Eigenschaften
Maschine mit SUSE Linux Enterprise Server 11 (64 Bit)	[OSType:UNIX] [OSType:LINUX] [OSType:LINUX][CPUArch:ppc64] [OSType:SUSE Linux Enterprise Server 11] [OSArch:64-bit]
Maschine mit Red Hat Enterprise Linux Server 6.3	[OSType:UNIX] [OSType:LINUX] [OSType:RedHat] [OSType:RedHatEnterpriseLinuxServer] [OSType:RedHatEnterpriseLinuxServer6.*]
Maschine mit SUSE Linux Enterprise Server 11 und der Umgebungsvariablen `@TPAE_DB_Server`, die auf `true` gesetzt ist	[OSType:UNIX] [OSType:LINUX] [OSType:SUSELinuxEnterpriseServer11][@TPAE_DB_Server:True]

Tabelle 8. Gescannte Abschnitte einer Konfigurationsdatei für UNIX (Forts.)

Betriebssysteme und Versionen	Abschnitte mit vorausgesetzten Eigenschaften

Maschine mit AIX 5.3.0.0 und der Umgebungsvariablen `@TPAE_DB_FEATURE` oder `@TPAE_DIR_FEATURE`, die auf `True` gesetzt ist	`[OSType:UNIX]`
	`[OSType:AIX]`
	`[OSType:AIX5.3.0.0][@TPAE_DB_FEATURE:True|@TPAE_DIR_FEATURE:True]`

Collector von Prerequisite Scanner

Die Collector von IBM Prerequisite Scanner erfassen basierend auf den vorausgesetzten Eigenschaften für die Produkte, die installiert werden sollen, Ist-Daten zur aktuellen Umgebung. Die Collector rufen die Daten über nativen Code ab. Bei den Daten kann es sich um allgemeine Daten, wie z. B. Daten zur Prozessorgeschwindigkeit und zum Arbeitsspeicher, Daten zur installierten Software, Betriebssystemdaten, Benutzerdaten, Netz-und Konnektivitätsdaten, handeln. Collector sind auch erweiterbar, d. h., Sie können angepasste Collector erstellen, um Ist-Werte für angepasste vorausgesetzte Eigenschaften abzurufen.

Prerequisite Scanner verwendet je nach Plattform Collector in den folgenden Sprachen:

- Windows: VBScript mit der Erweiterung `.vbs`
- UNIX: Shell mit der Erweiterung `.sh` oder ohne Erweiterung

Anmerkung: Sie können UNIX-Scripts nicht auf Windows-Systemen ausführen, selbst wenn Sie UNIX-ähnliche Umgebungen wie Cygwin auf den Windows-Maschinen installiert haben.

Collector für Windows-Systeme

VBScript-Collector für Windows-Systeme werden in der Windows-Script-Host-Umgebung ausgeführt. Sie verwenden Component Object Model, um auf Elemente der Windows-Umgebung wie `FileSystemObject` und `TextStream` zuzugreifen.

Prerequisite Scanner führt die VBScript-Collector aus, um die tatsächlichen Werte für die vorausgesetzten Eigenschaften für die Windows-Umgebung abzurufen. Jeder Collector kann Daten für eine oder mehrere vorausgesetzte Eigenschaften abrufen.

Für jede vorausgesetzte Eigenschaft in einem VBScript-Collector schreibt der Collector den Namen der vorausgesetzten Eigenschaft und ihren tatsächlichen Wert als Standardausgabe. Prerequisite Scanner schreibt diese Standardausgabe in eine temporäre Textdatei, d. h. `localhost_hw.txt`.

Sie können angepasste allgemeine VBScript-Collector erstellen, um Daten für vorausgesetzte Eigenschaften zu erfassen, die für alle Produkte und Produktversionen gelten. Sie können auch angepasste produktspezifische Collector erstellen, um Daten zu erfassen, die nur für ein bestimmtes Produkt und eine bestimmte Produktversion gelten.

Wenn Sie Prerequisite Scanner ausführen, werden die Collector in der folgenden Reihenfolge ausgeführt: vordefinierte VBScript-Collector, angepasste allgemeine VBScript-Collector im Verzeichnis *ips_root*`/lib` und angepasste produktspezifische VBScript-Collector, indem nach der Datei *product_code*[_<*version*>]`.vbs` im Verzeichnis *ips_root*`/Windows` gesucht wird.

Die Datei `env.tcrhome.vbs` ist beispielsweise ein angepasster Collector, der in dem mit der Umgebungsvariablen für das Ausgangsverzeichnis angegebenen Verzeichnis nach Tivoli Common Reporting sucht. Sie ist im Verzeichnis *ips_root*`/lib` gespeichert.

VBScript-Collector müssen den folgenden Regeln entsprechen:

- Namenskonvention für die angepasste allgemeine VBScript-Collectordatei

  Der Name enthält eine vorausgesetzte Eigenschaft, die jedem Produkt und jeder Produktversion, d. h. allen Konfigurationsdateien, bereitgestellt werden soll:

  prefix_identifier`.]property_name.vbs`

  Erläuterungen:

  – *prefix_identifier* ist die Präfix-ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich, zum Beispiel `env`.

  – *property_name* ist der Name der vorausgesetzten Eigenschaft, z. B. `tcrhome`.

  Speichern Sie diesen Typ von VBScript-Collector im Verzeichnis *ips_root*`/lib`.

- Namenskonvention für die angepasste produktspezifische VBScript-Collectordatei

  Der Name enthält Eigenschaften, die für ein bestimmtes Produkt und bestimmte Produktversionen verfügbar gemacht werden soll, d. h. eine Konfigurationsdatei:

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

  Erläuterungen:

  – *product_code*

    Dies ist die Variable für die Darstellung eines Produktcodes auf Windows- oder UNIX-Systemen. Produktcodes identifizieren das Produkt, eine individuelle Plattform wie Windows, AIX, HP-UX, Linux oder Solaris und optional die Version des Betriebssystems, das von diesem Produkt unterstützt wird. Sie werden in der Datei `codename.cfg` gespeichert. Jedes Produkt, das mehrere Plattformen unterstützt, hat mehrere Produktcodes, die jeweils ein Produkt, eine Plattform und eine Version des Betriebssystems identifizieren.

  – *<version>* ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100.

  Speichern Sie diesen Typ von VBScript-Collector im Verzeichnis *ips_root*`/Windows`.

- Die Standardausgabe für jede vorausgesetzte Eigenschaft ist wie folgt:

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

  – *property_name* stellt die vorausgesetzte Eigenschaft dar, die in der Konfigurationsdatei angegeben ist, z. B.:

    `env.tcrhome.`

  – *var_for_value*, d. h. die VBScript-Variable für den tatsächlichen Wert, den der Collector für die vorausgesetzte Eigenschaft abruft.

  Mit der folgenden Standardausgabe werden beispielsweise die vorausgesetzte Eigenschaft für die Umgebungsvariable für das Ausgangsverzeichnis von Tivoli Common Reporting und deren tatsächlicher Wert geschrieben:

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

Collector für UNIX-Systeme

Collector für UNIX-Systeme werden in der entsprechenden Shell-Host-Umgebung für AIX, HP-UX, Linux oder Solaris ausgeführt. Sie verwenden die speziellen Befehle und Optionen für diese Plattform, um auf Elemente der Hostumgebung zuzugreifen.

Jeder UNIX-Collector ruft Daten für eine vorausgesetzte Eigenschaft bzw. eine vorausgesetzte Eigenschaft mit vordefinierten Subtypen ab. Der Collector schreibt das Ergebnis der Prüfung für die vorausgesetzte Eigenschaft als Standardausgabe. Prerequisite Scanner schreibt diese Standardausgabe in eine temporäre Textdatei.

Sie können angepasste UNIX-Collector erstellen, um Daten für angepasste vorausgesetzte Eigenschaften zu erfassen. Jeder Collector, vordefiniert oder angepasst, wird in der Datei *ips_root*/UNIX_Linux/packageTest.sh aufgerufen.

Wenn Sie Prerequisite Scanner ausführen, werden die Collector in der folgenden Reihenfolge ausgeführt: vordefinierte Collector mit _plug im Dateinamen im Verzeichnis *ips_root*/lib, vordefinierte Collector im Verzeichnis *ips_root*/UNIX_Linux und angepasste UNIX-Collector im Verzeichnis *ips_root*/UNIX_Linux.

Die Datei installedSoftware.TCR.version ist beispielsweise ein angepasster Collector, der die Version von Tivoli Common Reporting abruft, die auf dem System installiert ist. Sie ist im Verzeichnis *ips_root*/UNIX_Linux gespeichert.

UNIX-Collector müssen den folgenden Regeln entsprechen:

- Namenskonvention für die angepasste UNIX-Collectordatei ohne Dateierweiterung:

  [*prefix_identifier*.]*property_name*

  Erläuterungen:

  – *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich, zum Beispiel installedSoftware.

  – *property_name* ist der Name der vorausgesetzten Eigenschaft, z. B. TCR.version.

  Speichern Sie den Collector im Verzeichnis *ips_root*/UNIX_Linux. Stellen Sie sicher, dass er keine Dateierweiterung hat.

- Standardausgabe für eine vorausgesetzte Eigenschaft, die den tatsächlichen Wert (Ist-Wert) für die vorausgesetzte Eigenschaft zurückgibt, wenn dieser eine ganze Zahl (Integer) oder eine Zeichenfolge (String) ist, z. B. die Softwareversion oder der verfügbare Plattenspeicherplatz für ein angehängtes Dateisystem. Alternativ kann der Collector "Unavailable" zurückgeben.

  ```
  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
  ```

- Code zum Aufrufen und Ausführen des Collectors im Script *ips_root*/UNIX_Linux/packageTest.sh:

```
res=`echo $line | grep installedSoftware.TCR.version`
if [ $res ]; then
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
```

Auswertungsprogramme von Prerequisite Scanner

Auswertungsprogramme von IBM Prerequisite Scanner sind Scripts, die die tatsächlichen Daten der Collector mit den erwarteten Daten für dieselben Eigenschaften, die in den Konfigurationsdateien enthalten sind, vergleichen. Auswertungen können plattformspezifisch sein, auf einfachen Operatoren wie kleiner als, gleich oder größer als oder darauf basieren, ob eine Eigenschaft installiert, vorhanden oder aktiviert ist. Sie können auch prüfen, ob Ports im Gebrauch oder verfügbar sind, und den Konnektivitätsstatus der Maschine feststellen. Sie können Auswertungsprogramme erstellen oder bearbeiten.

Prerequisite Scanner verwendet je nach Plattform Auswertungsprogramme in den folgenden Sprachen:

- Windows: VBScript mit der Erweiterung .vbs
- UNIX: Shell mit der Erweiterung .sh

Anmerkung: Sie können UNIX-Scripts nicht auf Windows-Systemen ausführen, selbst wenn Sie eine UNIX-ähnliche Umgebung auf den Windows-Maschinen wie Cygwin installiert haben.

Sie speichern Auswertungsprogramme in *ips_root*/*OS*, wobei *OS* für den Namen des Betriebssystems steht, z. B. Windows oder UNIX_Linux.

Auswertungsprogrammdateien müssen den folgenden Regeln entsprechen:

- Namenskonventionen für den Dateinamen:

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

  Erläuterungen:
  - *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
  - *property_name* ist der Name der vorausgesetzten Eigenschaft.
  - *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
- Übergeben Sie für komplexe Auswertungen optional zwei Eingabeparameter an das Script:
  - *expected_value*, d. h. der erwartete Wert für die vorausgesetzte Eigenschaft, die in der Konfigurationsdatei gesetzt ist
  - *actual_value*, d. h. der tatsächliche Wert, den der Collector für die vorausgesetzte Eigenschaft auf der Maschine erkennt

- Die Standardausgabe ist folgende:
  - `"PASS"`, wenn der erwartete Wert für die vorausgesetzte Eigenschaft größergleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist
  - `"FAIL"`, wenn der erwartete Werte für die vorausgesetzte Eigenschaft nicht gleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist

Ausgabeformate

IBM Prerequisite Scanner erzeugt Ausgaben für die folgenden Bildschirm- und lesbaren Dateiformate: Ausgabe in der Befehlszeilenschnittstelle, Debug- und Traceprotokolldateien, Text- und XML-Dateien für die Ergebnisse.

Prerequisite Scanner speichert die Scanergebnisse und Protokolldateien im Verzeichnis *ips_output_dir*. Sie können dieses Verzeichnis mit dem Eingabeparameter **`outputDir`** bei der Ausführung des Scanners angeben. Wenn Sie diesen Parameter nicht setzen, wird *ips_root* als Standardausgabeposition verwendet.

Anmerkung: Prerequisite Scanner erstellt während der Ausführung temporäre Dateien, jedoch werden diese vor Abschluss der Scannerausführung gelöscht. Diese temporären Dateien befinden sich im Unterverzeichnis *ips_output_dir*`/temp`. Der Scanner löscht auch das Unterverzeichnis *ips_output_dir*`/temp`, sofern das Unterverzeichnis keine Debug- und Tracedateien enthält, die ausschließlich auf UNIX-Systemen generiert wurden.

Sie können diesen Parameter auch verwenden, um eine Position anzugeben, wenn Sie Prerequisite Scanner von einer CD, einer DVD oder über ein schreibgeschütztes Netzlaufwerk ausführen.

Wichtig: Wenn das Ausgabeverzeichnis nicht vorhanden ist, erstellt Prerequisite Scanner das Verzeichnis. Sie müssen Schreibberechtigungen besitzen, um das Ausgabeverzeichnis, in dem Prerequisite Scanner die Dateien speichert, zu erstellen oder zu beschreiben.

Ausgabe in der Befehlszeilenschnittstelle

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **`detail`** setzen, zeigt Prerequisite Scanner detaillierte Ergebnisse des Scans in der Befehlszeilenschnittstelle an. Die detaillierten Ergebnisse enthalten Folgendes:

- Version von Prerequisite Scanner
- Version des Betriebssystems, unter dem der Scanner ausgeführt wurde
- Typ des Scans und Szenario
  - Scans nach Voraussetzungen: `Szenario: Scans nach Voraussetzungen`
- Namen der Produkte oder Komponenten, für die die Prüfungen der Voraussetzungen oder Diagnosen ausgeführt wurden
- Für jede vorausgesetzte Eigenschaft der Name der geprüften Eigenschaft, das Ergebnis PASS oder FAIL, der tatsächliche Wert und der erwartete Wert
- Für alle Komponenten der Name der allgemeinen Eigenschaft, das Ergebnis PASS oder FAIL, der tatsächliche Wert und der erwartete Wert
- Gesamtergebnis PASS oder FAIL (jeder Fehler einer Einzelprüfung führt zum Fehlschlagen des gesamten Scans)

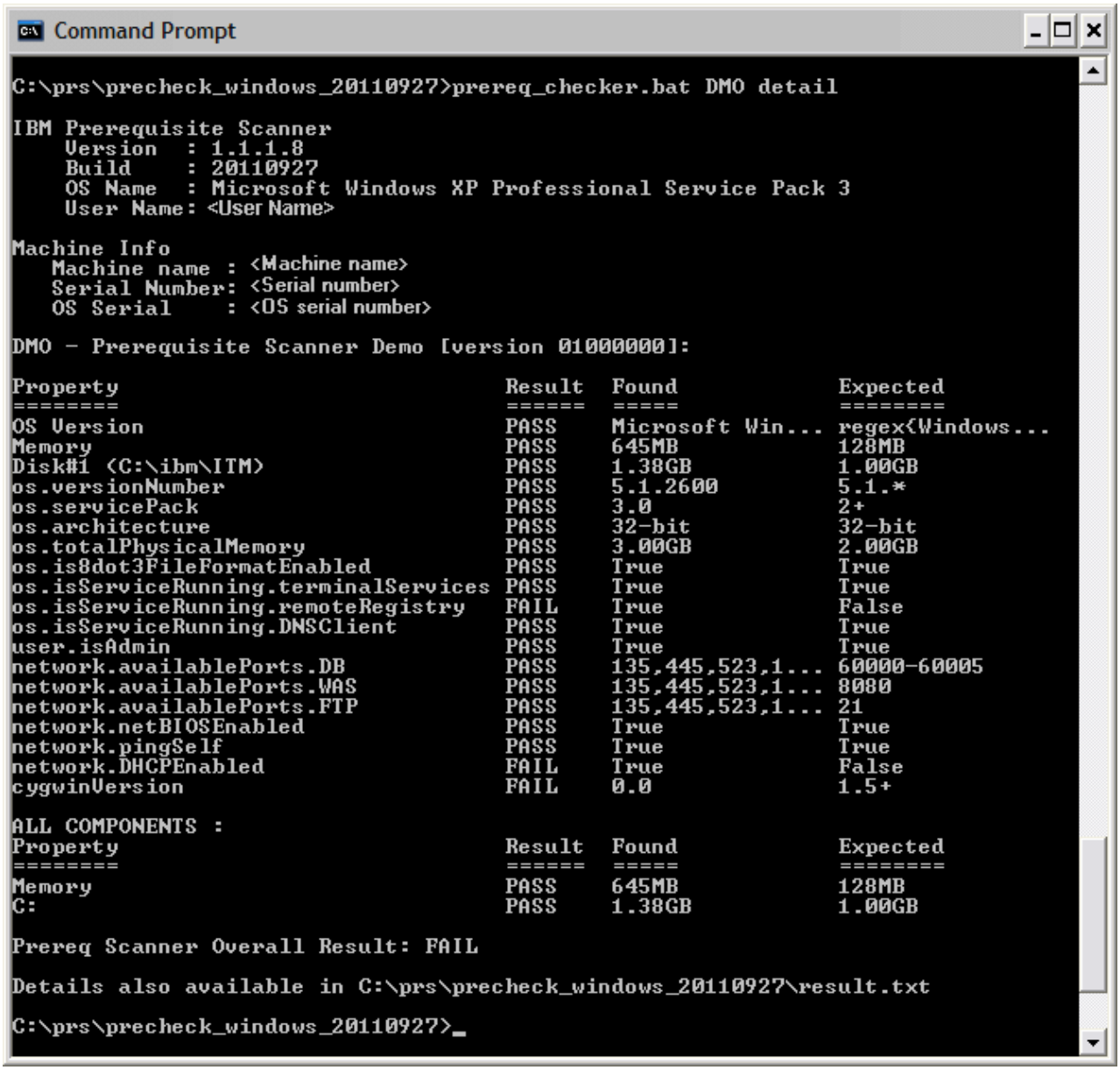


Abbildung 1. Ausgabe in der Befehlszeilenschnittstelle auf Windows-Systemen

Wenn Sie den Parameter **detail** nicht setzen, zeigt der Scanner nur das Ergebnis PASS oder FAIL am Bildschirm an.

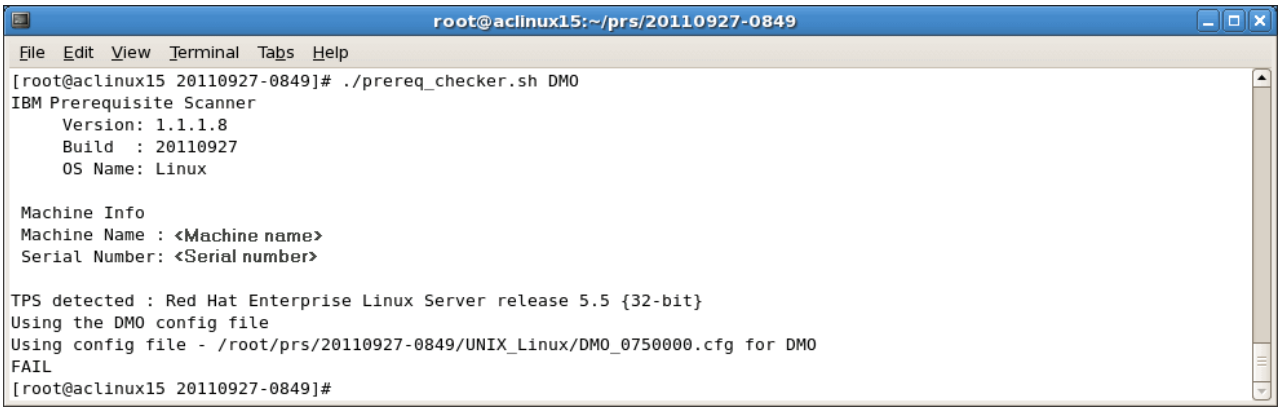


Abbildung 2. Ausgabe in der Befehlszeilenschnittstelle auf UNIX-Systemen

Prerequisite Scanner generiert Rückgabecodes, die von den Ergebnissen des Scans und davon abhängig sind, ob der Scanner aufgrund von Fehlern beendet werden muss. Diese Rückgabecodes werden in die Protokolldateien geschrieben. Wenn Prerequisite Scanner den Scan beispielsweise nicht ausführen kann, weil die Konfigurationsdatei nicht gelesen werden kann, wird der Rückgabecode 2 generiert.

Zusammenfassung der vorausgesetzten Eigenschaften für Hauptspeicher und Plattenspeicherplatz

Sie können Prerequisite Scanner ausführen, um die Voraussetzungen eines oder mehrerer Produkte oder einer oder mehrerer Komponenten gleichzeitig zu prüfen, indem Sie mehrere Produktcodes als Eingabeparameter angeben. Prerequisite Scanner fasst die Ergebnisse der Prüfung der Voraussetzungen für Hauptspeicher und Plattenspeicherplatz in den folgenden zusammengefassten Abschnitten der Ausgabe zusammen, sofern die vorausgesetzten Eigenschaften in einer Konfigurationsdatei angegeben wurden:

- Auf UNIX-Systemen im Abschnitt `TOTAL ALL SPECIFIED COMPONENTS`
- Auf Windows-Systemen im Abschnitt `ALL COMPONENTS`

Die allgemeinen vorausgesetzten Eigenschaften sind folgende:

- Physische Gesamthauptspeicherkapazität, die momentan in der Zielumgebung verfügbar ist:

  `Memory`
- Plattenspeicherplatz der Dateisysteme für die folgenden vorausgesetzten Eigenschaften:

Plattform	Vorausgesetzte Eigenschaften
UNIX und Linux	• `os.space.home` • `os.space.opt` • `os.space.tmp` • `os.space.usr` • `os.space.var`
Windows	`Disk`

Anmerkung: Wenn `opt`, `usr` und `var` nicht als Dateisysteme auf dem Zielcomputer definiert wurden, zeigt Prerequisite Scanner die erwarteten Werte nicht an und gibt Ergebnisse für diese vorausgesetzten Eigenschaften im zusammengefassten Abschnitt zurück.

Prerequisite Scanner zeigt den zusammengefassten Abschnitt nicht an, wenn in den Konfigurationsdateien weder vorausgesetzte Eigenschaften für den Hauptspeicher noch vorausgesetzte Eigenschaften für den Plattenspeicherplatz angegeben sind.

Prerequisite Scanner verarbeitet den Vergleich und die Anzeige von Plattenspeicherplatzwerten im zusammengefassten Abschnitt der Scanergebnisse anders als im Hauptabschnitt. Weitere Informationen finden Sie unter „Maßeinheiten in der Ausgabe" auf Seite 34.

Ausgabe in der Debugprotokolldatei auf Windows-Systemen

Prerequisite Scanner gibt Verarbeitungsinformationen, Warnungen und Fehlernachrichten sowie Scanergebnisse in der Datei *ips_output_dir*`/precheck.log` aus. Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **debug** setzen, gibt Prerequisite Scanner weitere Debugnachrichten in dieser Datei aus.

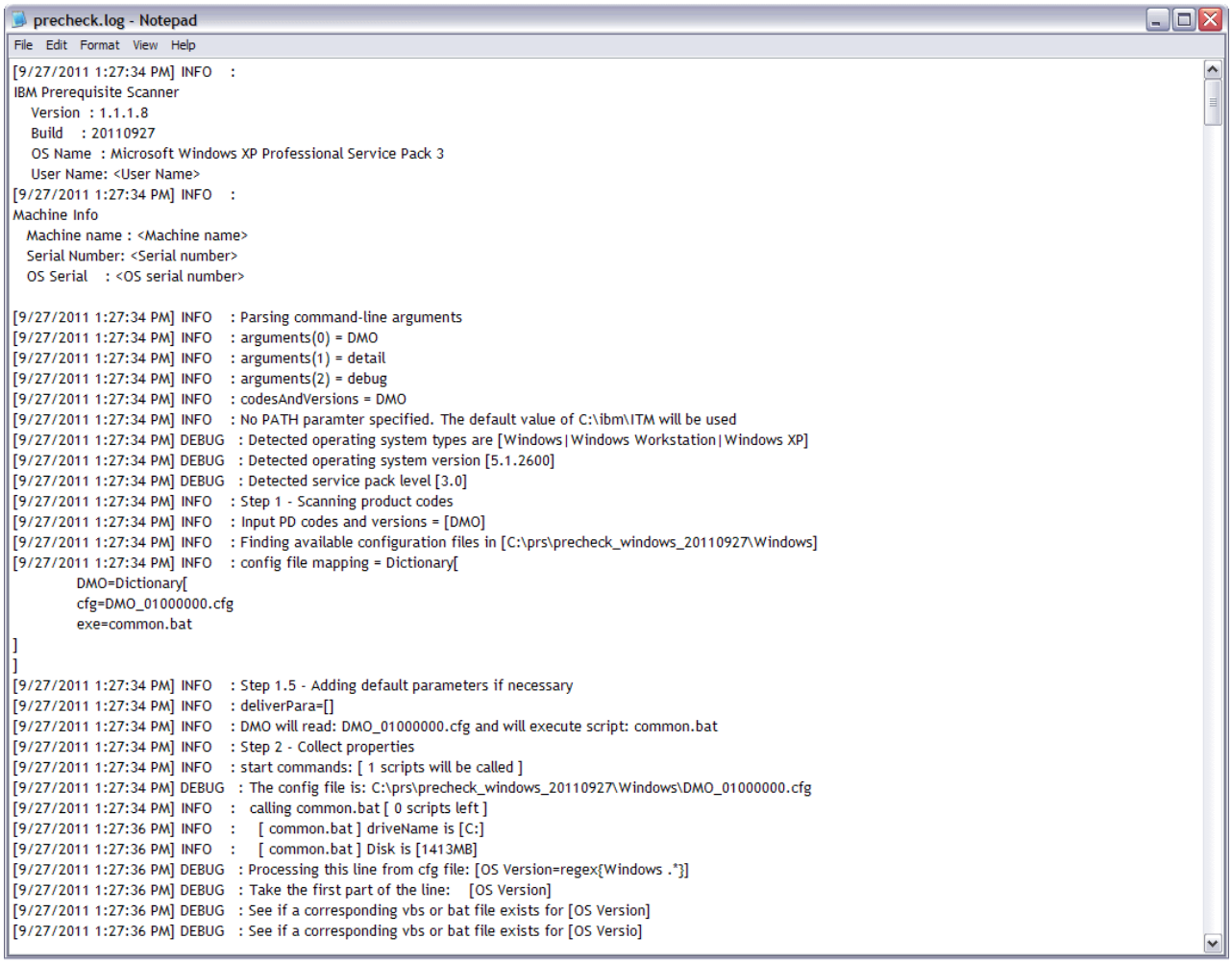


Abbildung 3. Datei "precheck.log"

Ausgabe in den Trace- und Debugprotokolldateien auf UNIX-Systemen

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **debug** setzen, gibt Prerequisite Scanner detaillierte Informationen, Warnungen und Fehlernachrichten zur Verarbeitung und die Scanergebnisse in der Datei *ips_output_dir*/temp/prs.debug aus.

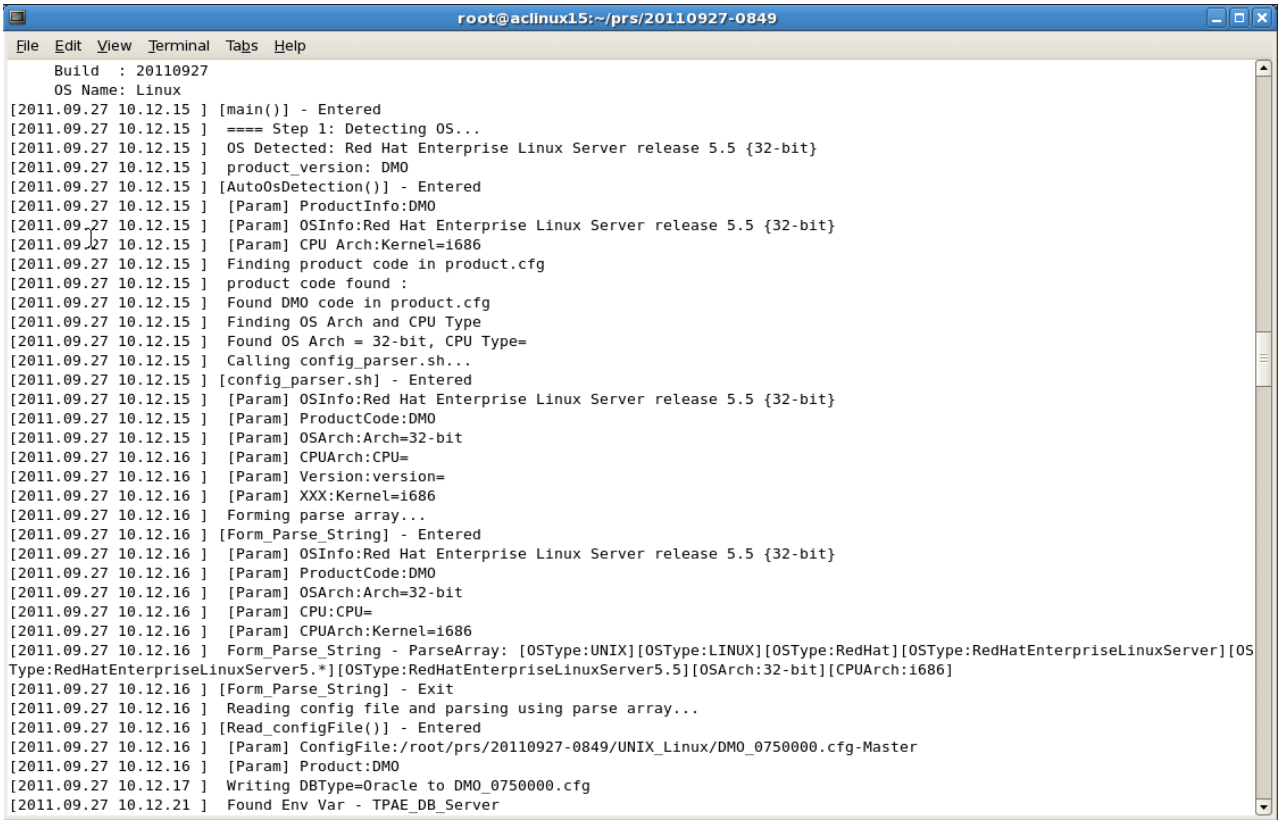


Abbildung 4. Datei "prs.debug" auf UNIX-Systemen

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **trace** setzen, gibt Prerequisite Scanner Traceinformationen in der Datei *ips_output_dir*/temp/prs.trc aus.

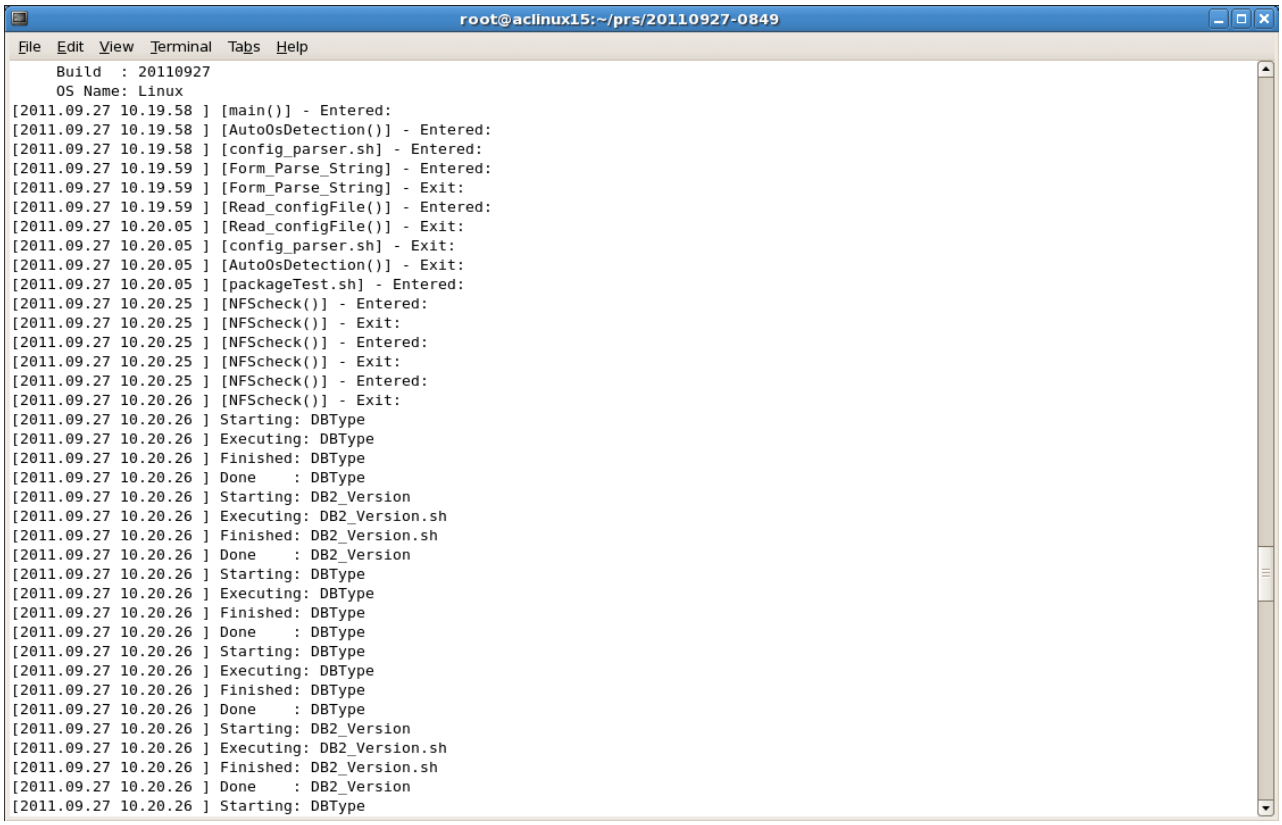
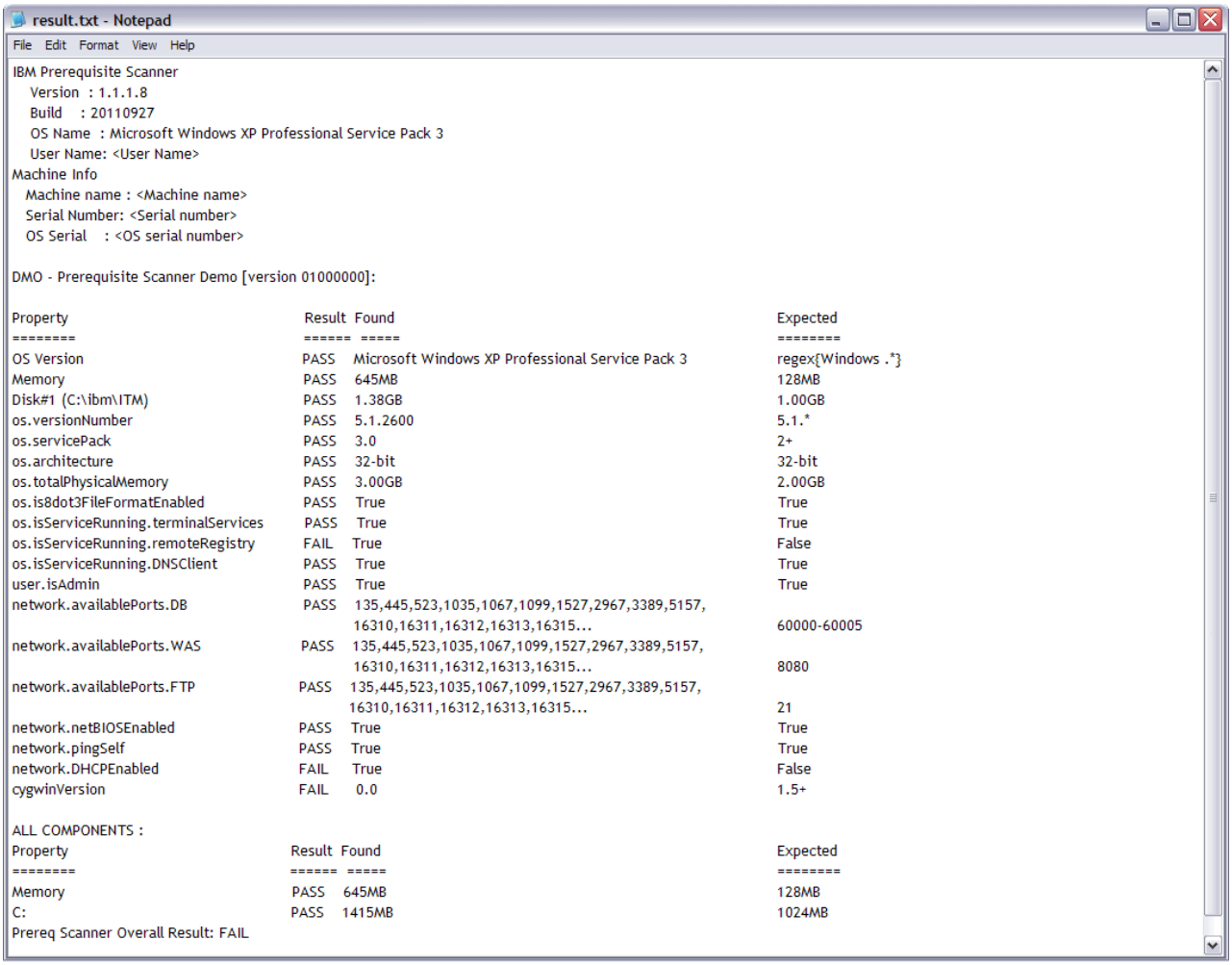


Abbildung 5. Datei "prs.trc" auf UNIX-Systemen

Ausgabe in einer Textdatei

Prerequisite Scanner gibt detaillierte Scanergebnisse in der Datei *ips_output_dir*/`result.txt` aus. Der Scanner speichert die Ergebnisse in der Textdatei unabhängig davon, ob Sie den Parameter **`detail`** setzen oder nicht.



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
```

Abbildung 6. Datei "result.txt" auf Windows-Systemen

```
[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
```

Abbildung 7. Datei "result.txt" auf UNIX-Systemen

Ausgabe in einer XML-Datei

Prerequisite Scanner gibt detaillierte Scanergebnisse in der Datei *ips_output_dir*/`result.xml` aus, wenn Sie den optionalen Eingabeparameter **`xmlResult`** setzen. Sie können diesen Parameter verwenden, um das Tool anzuweisen, die Ergebnisse nicht nur in der einfachen Testergebnisdatei, sondern auch in der XML-Ergebnisdatei auszugeben. Die Ergebnisse werden in der XML-Datei gespeichert, unabhängig davon, ob Sie den Parameter **`detail`** setzen oder nicht.

```
<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>
```

Abbildung 8. Datei "result.XML" auf Windows-Systemen

Entwickler können Java Developer Toolkit von Prerequisite Scanner für das Parsing und das Einlesen der XML-Datei verwenden.

Maßeinheiten in der Ausgabe

Prerequisite Scanner verarbeitet den Vergleich und die Anzeige von Plattenspeicherplatzwerten im zusammengefassten Abschnitt der Scanergebnisse anders als im Hauptabschnitt.

Im Hauptabschnitt der Scanergebnisse verarbeitet Prerequisite Scanner den Vergleich und die Anzeige der Plattenspeicherplatzwerte wie folgt:

Plattform	Vorausgesetzte Eigenschaften
UNIX und Linux	Wenn der tatsächliche Wert größer als 1024 MB ist, konvertiert und gibt Prerequisite Scanner den Wert in GB zurück. Andernfalls wird der Wert in MB zurückgegeben.

Plattform	Vorausgesetzte Eigenschaften
Windows	Prerequisite Scanner verwendet die Einheit des erwarteten Werts in der Konfigurationsdatei als Vergleichs- und Anzeigeeinheit. Gegebenenfalls wird der tatsächliche Wert in diese Einheit konvertiert.

Im zusammengefassten Abschnitt der Scanergebnisse verarbeitet Prerequisite Scanner den Vergleich und die Anzeige der Plattenspeicherplatzwerte wie folgt:

Plattform	Vorausgesetzte Eigenschaften
UNIX und Linux	Wenn der tatsächliche Wert größer als 1024 MB ist, konvertiert und gibt Prerequisite Scanner den Wert in GB zurück. Andernfalls wird der Wert in MB zurückgegeben.
Windows	Prerequisite Scanner führt den Vergleich der Plattenspeicherplatzwerte in MB-Einheiten durch. Für jede Konfigurationsdatei, die die vorausgesetzte Eigenschaft `Disk` enthält, konvertiert Prerequisite Scanner die tatsächlichen und erwarteten Werte in MB und führt dann den Vergleich durch. Wenn der zusammengefasste tatsächliche Wert größer als 1 GB ist, gibt Prerequisite Scanner den tatsächlichen Wert für Anzeigezwecke in GB mit der Genauigkeit 2 zurück. Andernfalls wird der Wert in MB zurückgegeben.

Java Developer Toolkit von Prerequisite Scanner

Java Developer Toolkit von Prerequisite Scanner stellt verschiedene APIs bereit, die Entwicklern ermöglichen, den Inhalt der XML-Ergebnisdatei für ihren Bedarf über das Programm zu parsen und einzulesen, z. B., um die Ergebnisse des Scans für die Verwendung in einem Installationsprogramm zu parsen.

Das Toolkit stellt die folgenden Pakete bereit:

- com.ibm.prs.common.exception

  Dieses Paket enthält die Klasse `PRSApiException`, die Methoden für das Auslösen von Ausnahmen für die XML-Abfrage-API bereitstellt.
- com.ibm.prs.common.reports.api

  Dieses Paket enthält die Schnittstelle `PRSXmlResultReader`, die die XML-Abfrage-API für die XML-Ergebnisdatei definiert.
- com.ibm.prs.common.reports.api.impl

  Dieses Paket enthält die Klasse `PRSXmlResultReaderImpl`, die `PRSXmlResultReader` implementiert.

Prerequisite Scanner kann die Formatierung und die Struktur anhand der XML-Schemadatei *ips_root*`/PRSResults.xsd` überprüfen.

Javadoc für das Toolkit wird im Verzeichnis *ips_root*`/api/javadoc` bereitgestellt.

XML-Schemadatei für die XML-Ergebnisdatei

Prerequisite Scanner stellt eine XML-Schemadatei bereit, anhand derer die XML-Ergebnisdatei validiert werden kann.

Die XML-Schemadateien enthalten die folgenden Elemente, die Abschnitte darstellen:

- `PRSInfo` für die Verwaltung der Dateils von Prerequisite Scanner
- `MachineInfo` für die Verwaltung von Informationen über die Zielumgebung, in der der Scan durchgeführt werden soll
- `UserInfo` für die Verwaltung von Informationen über den angemeldeten Benutzer, der den Scan durchführt
- `ScenarioInfo` für die Verwaltung des Typs von Scan und Szenario
- `ProductInfo` für die Verwaltung von Informationen über das Produkt bzw. die Komponente und die zugehörige Konfigurationsdatei
- `DetailedResults` für die Verwaltung der Scanergebnisse für jeden Satz vorausgesetzter Eigenschaften für ein Produkt oder eine Komponente, gruppiert nach `DetailedProductResultsElement`
- `AggregateResults` für die Verwaltung der zusammengefassten Scanergebnisse für den Plattenspeicherplatz und den Hauptspeicher
- `OverallResult` für die Verwaltung des Gesamtergebnisses (PASS oder FAIL) des Scans

Der Name und die Position des XML-Schemas sind *ips_root*`/PRSResults.xsd`.

Als Entwickler oder Implementierer können Sie Methoden aus der XML-Abfrage-API aufrufen, um die XML-Ergebnisdatei zu validieren. Javadoc für das Toolkit wird im Verzeichnis *ips_root*`/api/javadoc` bereitgestellt.

Scanvorgang

Wenn Sie IBM Prerequisite Scanner ausführen, führt der Scanner eine Reihe von Aufgaben in den einzelnen Migrationsstadien des Scanvorgangs aus. Der Benutzer öffnet eine Befehlszeilenschnittstelle und führt das Script von Prerequisite Scanner mit der Gruppe von Eingabeparametern, einschließlich eines Produktcodes, aus.

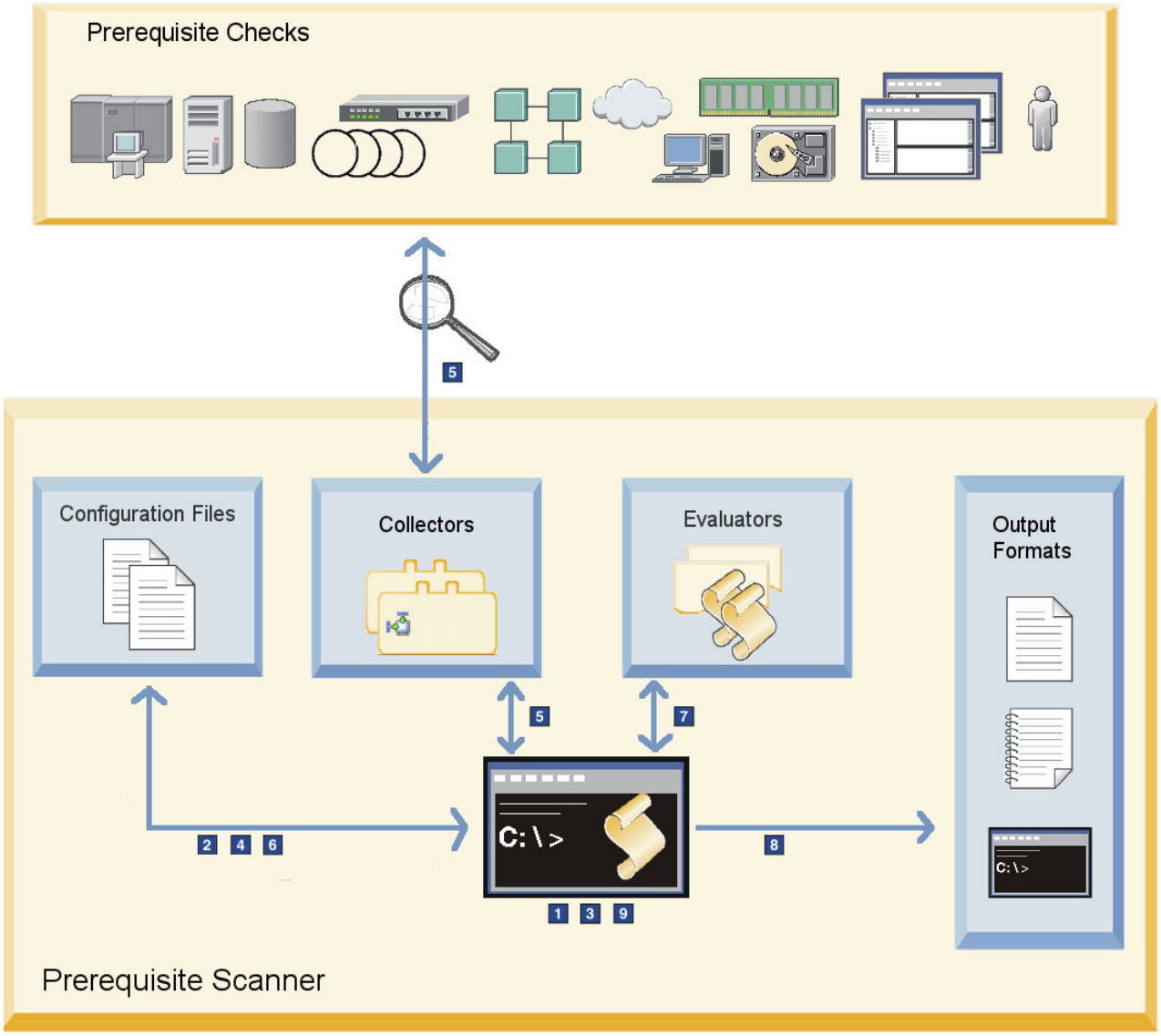


Abbildung 9. Architektur und Scanvorgang von Prerequisite Scanner

Im Folgenden finden Sie eine Zusammenfassung des Scanvorgangs in Abb. 9:

1. Prerequisite Scanner prüft das Format der Eingabeparameter, die an den Scanner übergeben werden.
2. Der Scanner bestimmt, ob der Produktcode, der als einer der Eingabeparameter übergeben wird, ein gültiger Produktcode in der Datei `codename.cfg` ist.
3. Der Scanner sucht die Konfigurationsdatei, die dem Produktcode zugeordnet ist. Wenn der optionale Parameter für die Produktversion nicht übergeben wurde, verwendet der Scanner die neueste Version der Konfigurationsdatei, die er im Verzeichnis *ips_root*`/Windows|UNIX_Linux` findet.
4. Der Scanner prüft, ob das tatsächliche Betriebssystem der Maschine ein unterstütztes Betriebssystem ist. Der Scanner überprüft das tatsächliche Betriebssystem anhand der erwarteten unterstützten Betriebssysteme in den Abschnittstiteln in der Konfigurationsdatei, deren Dateiname den Produktcode und die Produktversion enthält, die als Eingabeparameter übergeben wurden.

5. Der Scanner erfasst die tatsächlichen vorausgesetzten Eigenschaften für die Prüfungen der Voraussetzungen mithilfe der Collector von Prerequisite Scanner.
6. Der Scanner überprüft die vorausgesetzten Eigenschaften in der Konfigurationsdatei, die dem Produktcode und der Produktversion zugeordnet ist.

   Der Scanner überprüft das tatsächliche Betriebssystem anhand der erwarteten unterstützten Betriebssysteme in der vorausgesetzten Eigenschaft für die Betriebssystemversion (OS Version) und in den Abschnittstiteln in der Konfigurationsdatei, deren Dateiname den Produktcode und die Produktversion enthält, die als Eingabeparameter übergeben wurden.
7. Der Scanner liest die vorausgesetzten Eigenschaften aus der Konfigurationsdatei ein und analysiert die tatsächlichen und erwarteten Werte der vorausgesetzten Eigenschaften für die Prüfungen der Voraussetzungen. Bei Bedarf verwendet er die Auswertungsprogramme von Prerequisite Scanner.
8. Der Scanner gibt die Ergebnisse des Scans in der Befehlszeilenschnittstelle, in Ergebnistext- und -XML-Dateien sowie in lesbaren Protokolldateien aus.
9. Der Scanner bereinigt und entfernt temporäre Dateien und Verzeichnisse.

Neuerungen in diesem Release

IBM Prerequisite Scanner Version 1.2 stellt neue Eigenschaften und Erweiterungen bereit. Außerdem enthält das Produkt Fixes für Mängel.

Neue Features in dieser Modifikation

Möglichkeit zum Parsen und Einlesen der XML-Datei mit den neuen Scanergebnissen.

Java Developer Toolkit von Prerequisite Scanner stellt verschiedene APIs bereit, die Entwicklern ermöglichen, den Inhalt der XML-Ergebnisdatei für ihren Bedarf über das Programm zu parsen und einzulesen, z. B., um die Ergebnisse des Scans für die Verwendung in einem Installationsprogramm zu parsen. Weitere Informationen finden Sie unter „Java Developer Toolkit von Prerequisite Scanner" auf Seite 35.

Neue Konfigurationsdateien in dieser Modifikation

In Tabelle 9 sind die neuen Konfigurationsdateien und Produktcodes beschrieben, die mit Prerequisite Scanner Version 1.2 geliefert werden.

Tabelle 9. Neue Konfigurationsdateien

Produkt oder Komponente	Produktcode	Konfigurationsdatei
Tivoli Composite Application Manager Agent for WebSphere MQ	KMQ	*ips_root*/Windows|UNIX_Linux/KMQ_07010000.cfg
Tivoli Composite Application Manager Agent for WebSphere Message Broker	KQI	*ips_root*/Windows|UNIX_Linux/KQI_07010000.cfg

Neue vorausgesetzte Eigenschaften in dieser Modifikation

Die Eigenschaft `os.SeaMonkeyVersion` wurde hinzugefügt, damit die Version von Mozilla SeaMonkey auf der Maschine geprüft werden kann. Weitere Informationen finden Sie unter „Dateneigenschaften für Betriebssysteme" auf Seite 101.

Die Eigenschaft `env.var.set.`*`env_var_name`* wurde hinzugefügt, um zu prüfen, ob die mit *env_var_name* angegebene Umgebungsvariable auf der Maschine gesetzt ist. Weitere Informationen finden Sie unter „Dateneigenschaften für Umgebungsvariablen" auf Seite 115.

Erweiterungen in dieser Modifikation

Möglichkeit, Scanergebnisse in eine XML-Datei zu schreiben.

Die Datei *ips_output_dir*/`result.xml` ist die Datei, die die neuen Scanergebnisse im XML-Format enthält. Standardmäßig gibt das Tool die Ergebnisse nur in einer einfachen Textergebnisdatei aus. Weitere Informationen finden Sie unter „Ausgabeformate" auf Seite 26.

`xmlResult` ist ein neuer, optionaler Eingabeparameter für das Script von Prerequisite Scanner in Prerequisite Scanner Version 1.2. Sie können diesen Parameter verwenden, um das Tool anzuweisen, die Ergebnisse nicht nur in der einfachen Testergebnisdatei, sondern auch in der XML-Ergebnisdatei auszugeben. Weitere Informationen finden Sie unter „prereq_checker" auf Seite 67.

Entfernen der zusammengefassten Abschnitte in den Ergebnissen, wenn weder vorausgesetzte Eigenschaften für den Hauptspeicher noch vorausgesetzte Eigenschaften für den Plattenspeicherplatz in den Konfigurationsdateien angegeben sind.
Prerequisite Scanner zeigt die zusammengefassten Abschnitte in der Ergebnisdatei nicht mehr an, wenn weder vorausgesetzte Eigenschaften für den Hauptspeicher noch vorausgesetzte Eigenschaften für den Plattenspeicherplatz in den Konfigurationsdateien angegeben sind. Weitere Informationen finden Sie unter „Ausgabeformate" auf Seite 26.

Veraltete Features in dieser Modifikation

Keine

In dieser Modifikation behobene Mängel

Eine Liste der in diesem Release behobenen Mängel finden Sie in der Datei `Readme.html` im Verzeichnis *ips_root*, wenn Sie den Inhalt der Softwarepakete von Prerequisite Scanner extrahieren.

Dokumentationsänderungen in dieser Modifikation

Das Benutzerhandbuch zu Prerequisite Scanner ist nicht mehr mit den plattformspezifischen Paketen von Prerequisite Scanner gebündelt. Sie können das Information Center von IBM Prerequisite Scanner verwenden.

Kapitel 2. Prerequisite Scanner installieren

Es gibt kein Installationsprogramm für IBM Prerequisite Scanner. Wenn Sie den Inhalt der komprimierten Datei extrahieren, befinden sich die Kerndateien im Stammverzeichnis, das die folgenden Unterverzeichnissen enthält: "/api für das Java Developer Toolkit von Prerequisite Scanner für die Unterstützung der Abfrage-XML-API, /lib für die Collector und allgemeinen Scripts, /Windows für die Auswertungsprogramme und Konfigurationsdateien unter Windows, /UNIX_Linux für die Auswertungsprogramme und Konfigurationsdateien auf UNIX-Plattformen und /licenses für die Lizenzdateien.

Voraussetzungen

IBM Prerequisite Scanner kann auf Windows-Systemen, Windows XP und höher, 32 Bit oder 64 Bit ausgeführt werden. Der Scanner kann auch unter Varianten der Betriebssysteme AIX, HP-UX, Linux und Solaris ausgeführt werden.

Stellen Sie sicher, dass die folgenden Dienstprogramme in den Zielumgebungen installiert bzw. verfügbar sind:

Zielsystem	Voraussetzungen
Windows	• Der Telnet-Client ist aktiviert, damit Konnektivitätsprüfungen im vordefinierten Konnektivitätscollector ordnungsgemäß funktionieren. • Microsoft .Net Framework 1.0 oder höher ist installiert, damit Prerequisite Scanner ordnungsgemäß funktioniert.

Zielsystem	Voraussetzungen

UNIX	• Bash ist installiert, damit die UNIX-Collector von Prerequisite Scanner ordnungsgemäß funktionieren.
	• Für Benutzer ohne Rootberechtigung muss die Position der Befehle **mount**, **swapinfo** und **psrinfo** in der Umgebungsvariablen PATH gesetzt werden, damit die Befehle für Prerequisite Scanner verfügbar sind. Sind die Befehle beispielsweise im Verzeichnis /usr/sbin enthalten, setzen Sie die Umgebungsvariable PATH wie folgt:
	`export PATH=$PATH:/usr/sbin/`
	• Stellen Sie sicher, dass dem Befehl **lscfg** die richtigen Zugriffsberechtigungen zugeordnet werden, einschließlich aller speziellen Berechtigungen, die über die Flags für Zugriffsrechte wie das setuid-Bit gesetzt werden. Die richtigen Zugriffsberechtigungen gewährleisten, dass Prerequisite Scanner den Befehl ausführen und die Systeminformationen abrufen kann. Wenn sich der Befehl beispielsweise im Verzeichnis /usr/sbin befindet, müssen Sie zum Setzen des setuid-Bits für **lscfg** den Befehl **chmod** wie folgt ausführen:
	`chmod 4777 /usr/sbin/lscfg`

Prerequisite Scanner unterstützt alle Hardware- und Betriebssysteme des angegebenen Produkts bzw. der IBM Lösung, für das bzw. die Sie Prerequisite Scanner ausführen.

Komprimierte Datei installieren

Sie können den Inhalt der komprimierten Datei für IBM Prerequisite Scanner extrahieren. Sie müssen Schreibberechtigungen für das Stammverzeichnis haben, in das Sie den Inhalt der komprimierten Datei extrahieren.

Vorgehensweise

1. Öffnen Sie Ihren Web-Browser, und geben Sie den URL für IBM Fix Central ein. Stellen Sie sicher, dass Sie sich bei IBM.com oder IBM Support Portal anmelden.
2. Wählen Sie in der Liste **Product Group** den Eintrag **Tivoli** aus.
3. Wählen Sie in der Liste **Product** den Eintrag IBM Prerequisite Scanner aus.
4. Wählen Sie in der Liste **Installed Version** die Version aus, die Sie herunterladen möchten.
5. Wählen Sie in der Liste **Platform** die Plattform aus, auf der Sie Prerequisite Scanner installieren möchten.
6. Klicken Sie auf **Continue**. Die Seite **Identify Fixes** wird geöffnet.
7. Verwenden Sie die Standardoption, **Browse for fixes**, und klicken Sie auf **Continue**.

8. Wählen Sie auf der Seite **Select fixes** das Paket aus, und klicken Sie auf **Continue**.
9. Wählen Sie auf der Seite **Download Option** die Downloadoption aus, und klicken Sie auf **Download now**.
10. Extrahieren Sie den Inhalt der komprimierten Datei an die bevorzugte Position, die mit *ips_root* angegeben wurde.

Nächste Schritte

Lesen Sie in der Installationsdokumentation zum Produkt oder in den technischen Hinweisen nach, welche zusätzlichen Schritte vor der Ausführung von Prerequisite Scanner ausgeführt werden müssen. Möglicherweise müssen Sie beispielsweise die Umgebungsvariable setzen, die Prerequisite Scanner anzeigt, welche Komponenten oder Features auf dem Zielsystem installiert werden, und damit, welche Voraussetzungen zu prüfen sind.

Prerequisite Scanner deinstallieren

Entfernen Sie IBM Prerequisite Scanner, wenn Sie eine neuere Version installieren möchten, das Programm in eine andere Umgebung versetzen möchten oder diese Version nicht mehr benötigen.

Vorgehensweise

1. Öffnen Sie das Verzeichnis *ips_root*.
2. Löschen Sie das Verzeichnis und dessen Inhalt.

Kapitel 3. Prerequisite Scanner erweitern

IBM Prerequisite Scanner stellt eine Reihe von Collector, Auswertungsprogrammen und Konfigurationen bereit, die Sie verwenden können, um das Tool und den Scan für die Voraussetzungen auszuführen. Wenn die Basisgruppe von Dateien, die vorausgesetzten Eigenschaften und deren Werte und die Prüfungen der Voraussetzungen nicht Ihren Anforderungen entsprechen, können Sie Prerequisite Scanner erweitern.

Vorbereitungen für die Ausführung von Prerequisite Scanner

Bestimmen Sie vor der Ausführung von IBM Prerequisite Scanner, ob die vordefinierten vorausgesetzten Eigenschaften, deren erwartete Werte und die Konfigurationsdateien Ihren Anforderungen bezüglich des Scans der Voraussetzungen entsprechen. Wenn diese nicht Ihren Anforderungen entsprechen, können Sie eine Reihe von Aufgaben für Voraussetzungen ausführen, um Prerequisite Scanner zu konfigurieren oder zu erweitern. Die Gruppe der Prüfungen und Aufgaben für Voraussetzungen richtet sich nach der Plattform und der Anzahl der Prüfungen der Voraussetzungen.

Erforderliche Prüfungen und Erweiterungsaufgaben für Windows-Systeme

Sie müssen verschiedene Prüfungen und Aufgaben ausführen, bevor Sie IBM Prerequisite Scanner ausführen. Diese Prüfungen stellen fest, ob Sie vorhandene Konfigurationsdateien bearbeiten und verwenden können oder ob Sie Prerequisite Scanner erweitern müssen.

Tabelle 10 stellt eine Liste mit den auszuführenden Prüfungen und Aufgaben bereit.

Tabelle 10. Prüfungen und Aufgaben vor der Verwendung einer Konfigurationsdatei für Windows-Systeme

	Prüfung	**Aufgabe**
☐	Prüfen, ob das Produkt, die unterstützten Betriebssysteme und die Versionen des Betriebssystems in der Datei `codename.cfg` aufgelistet sind.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, Produktcode für das Produkt, das jeweilige Betriebssystem und die optionale Betriebssystemversion in der Datei hinzufügen. Weitere Informationen hierzu finden Sie im Abschnitt „Produktcodes hinzufügen" auf Seite 47.
☐	Prüfen, ob eine Konfigurationsdatei für den Produktcode, der der Produktversion zugeordnet ist, vorhanden ist.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, Konfigurationsdatei mit den vorausgesetzten Eigenschaften für dieses Betriebssystem und die Version des Betriebssystems erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Konfigurationsdateien erstellen" auf Seite 48.
☐	Konfigurationsdatei öffnen und prüfen, ob sie die richtigen vorausgesetzten Eigenschaften enthält.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, vorausgesetzte Eigenschaften hinzufügen. Weitere Informationen finden Sie im Abschnitt „Vorausgesetzte Eigenschaften hinzufügen" auf Seite 50.

Tabelle 10. Prüfungen und Aufgaben vor der Verwendung einer Konfigurationsdatei für Windows-Systeme (Forts.)

	Prüfung	Aufgabe
☐	Prüfen, ob die vorausgesetzten Eigenschaften die erwarteten Werte haben.	• Wenn ja, Prerequisite Scanner ausführen. Weitere Informationen finden Sie in Kapitel 4, „Prerequisite Scanner ausführen", auf Seite 67. • Wenn nicht, vorausgesetzte Eigenschaften bearbeiten. Weitere Informationen finden Sie im Abschnitt „Vorausgesetzte Eigenschaften bearbeiten" auf Seite 52.
☐	Für alle neuen vorausgesetzten Eigenschaften prüfen, ob vordefinierte Collector die tatsächlichen Werte für die vorausgesetzten Eigenschaften erfassen können.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, angepasste Collector erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Collector für Windows-Systeme erstellen" auf Seite 53.
☐	Für alle neuen oder bearbeiteten vorausgesetzten Eigenschaften prüfen, ob vordefinierte Auswertungsprogramme die erwarteten und tatsächlichen Werte für die vorausgesetzten Eigenschaften vergleichen können.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, angepasste Auswertungsprogramme erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Auswertungsprogramme für Windows-Systeme erstellen" auf Seite 60.
☐	Sicherstellen, dass alle Dateien in den richtigen Verzeichnissen gespeichert wurden. • Konfigurationsdateien, angepasste produktspezifische Collector und zugehörige Stapeldateien sowie angepasste Auswertungsprogrammdateien im Verzeichnis *ips_root*`/Windows` • Angepasste allgemeine Collector im Verzeichnis *ips_root*`/lib`	Prerequisite Scanner ausführen. Weitere Informationen finden Sie in Kapitel 4, „Prerequisite Scanner ausführen", auf Seite 67.

Erforderliche Prüfungen und Erweiterungsaufgaben für UNIX-Systeme

Sie müssen verschiedene Prüfungen der Voraussetzungen und Aufgaben ausführen, bevor Sie IBM Prerequisite Scanner ausführen. Diese Prüfungen stellen fest, ob Sie vorhandene Konfigurationsdateien bearbeiten und verwenden können oder ob Sie Prerequisite Scanner erweitern müssen.

Tabelle 11 stellt eine Liste mit den auszuführenden erforderlichen Prüfungen und Aufgaben bereit.

Tabelle 11. Prüfungen und Aufgaben vor der Verwendung einer Konfigurationsdatei für UNIX-Systeme

	Prüfung	Aufgabe
☐	Prüfen, ob das Produkt in der Datei `codename.cfg` aufgelistet ist.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, Produktcode in der Datei `codename.cfg` hinzufügen. Weitere Informationen hierzu finden Sie im Abschnitt „Produktcodes hinzufügen" auf Seite 47.

Tabelle 11. Prüfungen und Aufgaben vor der Verwendung einer Konfigurationsdatei für UNIX-Systeme (Forts.)

	Prüfung	Aufgabe
☐	Prüfen, ob eine Konfigurationsdatei für den Produktcode, der dem Produkt zugeordnet ist, vorhanden ist.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, Konfigurationsdatei mit den vorausgesetzten Eigenschaften für alle unterstützten Plattformen des Produkts erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Konfigurationsdateien erstellen" auf Seite 48.
☐	Konfigurationsdatei öffnen und prüfen, ob sie die richtigen vorausgesetzten Eigenschaften enthält.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, vorausgesetzte Eigenschaften hinzufügen. Weitere Informationen finden Sie im Abschnitt „Vorausgesetzte Eigenschaften hinzufügen" auf Seite 50.
☐	Prüfen, ob die vorausgesetzten Eigenschaften die erwarteten Werte haben.	• Wenn ja, Prerequisite Scanner ausführen. Weitere Informationen finden Sie in Kapitel 4, „Prerequisite Scanner ausführen", auf Seite 67. • Wenn nicht, vorausgesetzte Eigenschaften bearbeiten. Weitere Informationen finden Sie im Abschnitt „Vorausgesetzte Eigenschaften bearbeiten" auf Seite 52.
☐	Für alle neuen vorausgesetzten Eigenschaften prüfen, ob vordefinierte Collector die tatsächlichen Werte für die vorausgesetzten Eigenschaften erfassen können.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, angepasste Collector erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Collector für UNIX-Systeme erstellen" auf Seite 57.
☐	Für alle neuen oder bearbeiteten vorausgesetzten Eigenschaften prüfen, ob Auswertungsprogramme die erwarteten und tatsächlichen Werte für die vorausgesetzten Eigenschaften vergleichen können.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, angepasste Auswertungsprogramme erstellen. Weitere Informationen finden Sie im Abschnitt „Angepasste Auswertungsprogramme für UNIX-Systeme erstellen" auf Seite 65.
☐	Für alle neuen oder bearbeiteten Eigenschaften prüfen, ob der Code für den Aufruf und die Ausführung der Collector im Script *ips_root*/UNIX_Linux/packageTest.sh enthalten ist.	• Wenn ja, nächste Prüfung durchführen. • Wenn nicht, Masterpakettestscript bearbeiten. Weitere Informationen finden Sie im Abschnitt „Pakettestscript für UNIX-Systeme bearbeiten" auf Seite 59.
☐	Sicherstellen, dass alle Dateien in den richtigen Verzeichnissen gespeichert wurden. • Konfigurationsdateien, angepasste Collectordateien und angepasste Auswertungsprogrammdateien im Verzeichnis *ips_root*/UNIX_Linux	Prerequisite Scanner ausführen. Weitere Informationen finden Sie in Kapitel 4, „Prerequisite Scanner ausführen", auf Seite 67.

Produktcodes hinzufügen

IBM Prerequisite Scanner stellt eine Reihe vordefinierter Produktversionscodes in der Datei codename.cfg bereit. Sie können Produktcodes hinzufügen, wenn diese für die Produktversion, die unterstützten Plattformen und die Versionen der Betriebssysteme nicht in der Datei enthalten sind.

Vorgehensweise

1. Öffnen Sie die Datei *ips_root*/codename.cfg.
2. Überprüfen Sie, ob die Datei bereits Name/Wert-Paare für die Produktversionen enthält.

3. Wenn der Produktcode nicht vorhanden ist, fügen Sie einen hinzu, und stellen Sie sicher, dass Sie das richtige Format verwenden, das im Folgenden gezeigt wird:

   product_code=code_value

   Einschränkung: IBM Tivoli Monitoring und Tivoli Composite Application Manager haben vordefinierte Produktcodes, die Prerequisite Scanner als reservierte Codes betrachtet. Diese Codes dürfen nicht als Produktcodes von Prerequisite Scanner verwendet werden, es sei denn, sie verweisen auf die zugehörigen Agenten von IBM Tivoli Monitoring und Tivoli Composite Application Manager. Weitere Informationen zum Hinzufügen der Produktcodes finden Sie im technischen Hinweis zu den Produktcodes für ITM 6.X.

   Einschränkung: Nur UNIX: Wenn Sie den Wert für den Produktcode in der Datei eingeben, vermeiden Sie die Verwendung von `for`. Dies ist ein reserviertes Wort und kann sich auf die Ausführung von Prerequisite Scanner auswirken.

   Wenn Sie beispielsweise einen Produktcode für IBM Tivoli Monitoring for Energy Management auf allen Windows-Plattformen hinzufügen möchten, fügen Sie der Datei die folgende Zeile hinzu:

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

Angepasste Konfigurationsdateien erstellen

Sie können angepasste Konfigurationsdateien aus der Beispielkonfigurationsdatei erstellen, wenn die vordefinierten Konfigurationsdateien nicht Ihren Anforderungen für die vorausgesetzten Eigenschaften entsprechen. Bevor Sie die angepasste Konfigurationsdatei erstellen, stellen Sie sicher, dass Sie die vorausgesetzten Eigenschaften, die Sie hinzufügen möchten, und deren erwartete Werte kennen.

Informationen zu diesem Vorgang

Wichtig: Sie müssen die Namenskonventionen und Formatierungsregeln für das Erstellen und Bearbeiten einer angepassten Konfigurationsdatei einhalten. Wenn Sie dies nicht tun, kann Prerequisite Scanner keinen erfolgreichen Scan mit dieser Datei durchführen.

Vorgehensweise

1. Fügen Sie bei Bedarf Produktcodes für das Produkt in der Datei `codename.cfg` hinzu.
2. Erstellen Sie die Konfigurationsdatei mithilfe eines Texteditors im Verzeichnis *ips_root*`/OS`. Stellen Sie sicher, dass Sie die folgende Namenskonvention für den Dateinamen verwenden:

   product_code_version`.cfg`

   Erläuterungen:

   - *product_code*

     Dies ist die Variable für die Darstellung eines Produktcodes auf Windows- oder UNIX-Systemen. Produktcodes identifizieren das Produkt, eine individuelle Plattform wie Windows, AIX, HP-UX, Linux oder Solaris und optional die Version des Betriebssystems, das von diesem Produkt unterstützt wird. Sie werden in der Datei `codename.cfg` gespeichert. Jedes Produkt, das mehrere Plattformen unterstützt, hat mehrere Produktcodes, die jeweils ein Produkt, eine Plattform und eine Version des Betriebssystems identifizieren.

- *version* ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100.

3. Überprüfen Sie die vorausgesetzten Basiseigenschaften, die in Anhang C, „Referenzinformationen zu vorausgesetzten Eigenschaften", auf Seite 91 beschrieben sind, und bestimmen Sie, welche vorausgesetzten Eigenschaften Sie überprüfen möchten.
4. Optional: Fügen Sie einen Abschnitt hinzu, und stellen Sie sicher, dass Sie die folgende Namenskonvention für die Abschnittstitel verwenden:
   - **Einzelne, vordefinierte Datentypkategorie**

     [*category_name*:*category_value*]

     Wenn Sie beispielsweise einen Abschnitt für vorausgesetzte Eigenschaften, die für alle Windows-Plattformen gelten, erstellen möchten, fügen Sie den folgenden Abschnittstitel hinzu:

     ```
     [OSType:Windows]
     ```

     Wenn Sie beispielsweise einen Abschnitt für vorausgesetzte Eigenschaften, die für alle Linux-Betriebssystemvarianten gelten, erstellen möchten, fügen Sie den folgenden Abschnittstitel hinzu:

     ```
     [OSType:RedHat]
     ```

   - **Kombinierte, vordefinierte Datentypkategorien**

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

     Wenn Sie beispielsweise einen Abschnitt für vorausgesetzte Eigenschaften, die für alle Varianten von Windows Server 2003, ausschließlich Windows Server 2003 R2, gelten, erstellen möchten, fügen Sie den folgenden Abschnittstitel hinzu:

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

     Wenn Sie beispielsweise einen Abschnitt für vorausgesetzte Eigenschaften erstellen möchten, die für SUSE Linux Enterprise Server 11 OS gelten und davon abhängig sind, ob die Umgebungsvariable `@TPAE_DB_SERVER` auf `true` gesetzt ist, fügen Sie den folgenden Abschnittstitel hinzu:

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

   Erläuterungen:

   category_name ist der Mehrzeichencode, der die Datentypkategorie darstellt. Weitere Informationen hierzu finden Sie in Tabelle 6 auf Seite 17.

   category_value ist der Mehrzeichencode, der einen zulässigen Wert für die Kategorie darstellt. Weitere Informationen hierzu finden Sie in Tabelle 6 auf Seite 17.
5. Optional: Überprüfen Sie für jeden Abschnitt die vorausgesetzten Basiseigenschaften, die in Anhang C, „Referenzinformationen zu vorausgesetzten Eigenschaften", auf Seite 91 beschrieben sind, und bestimmen Sie, welche vorausgesetzten Eigenschaften Sie überprüfen möchten.
6. Geben Sie für jede vorausgesetzte Eigenschaft, die Sie hinzufügen möchten, das Name/Wert-Paar mit den ggf. erforderlichen optionalen Qualifikationsmerkmalen ein. Stellen Sie sicher, dass Sie das folgende Format verwenden, und geben Sie jede vorausgesetzte Eigenschaft in jeweils einer Zeile an:

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

   Erläuterungen:

- *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
- *property_name* ist der Name der vorausgesetzten Eigenschaft.
- *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
- *qualifier_name* ist ein optionales Attribut für die vorausgesetzte Eigenschaft. IBM Prerequisite Scanner verwendet dieses Attribut, um die vorausgesetzte Eigenschaft bzw. den Typ der für die vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren. Weitere Informationen finden Sie im Abschnitt „Vordefinierte Qualifikationsmerkmale für vorausgesetzte Eigenschaften" auf Seite 9.

  Anmerkung: Sie können mehrere Qualifikationsmerkmale durch Kommas getrennt angeben. Die Gruppe der Qualifikationsmerkmale muss in eckige Klammern (`[]`) eingeschlossen werden.
- *qualifier_value* ist der Wert für das optionale Attribut. Jedes Qualifikationsmerkmal und dessen Wert muss durch einen Doppelpunkt (`:`) begrenzt werden.
- *property_value* ist der Wert für die vorausgesetzte Eigenschaft und kann eine Zeichenfolge (String) oder eine ganze Zahl (Integer) sein.

Die benutzerdefinierte Kategorie der vorausgesetzten Eigenschaften hat beispielsweise die Präfix-ID `user`. Die vorausgesetzte Eigenschaft für die Überprüfung, ob der angemeldete Benutzer zur Benutzergruppe der Administratoren gehört, ist `user.isAdmin=True`.

7. Wenn eine vorausgesetzte Eigenschaft nicht in den vordefinierten Kategorien vorhanden ist, fügen Sie den Namen, den Wert und optionale Qualifikationsmerkmale für die angepasste vorausgesetzte Eigenschaft hinzu. Anschließend müssen Sie die folgenden Dateien erstellen, um die angepasste vorausgesetzte Eigenschaft zu suchen und bei Bedarf zu vergleichen: einen angepassten Collector für die Erfassung des tatsächlichen Werts für die angepasste Eigenschaft und ein angepasstes Auswertungsprogramm, wenn die Standardvergleichsfunktionen für den Vergleich des tatsächlichen Werts mit dem erwarteten Wert nicht funktionieren.

Vorausgesetzte Eigenschaften hinzufügen

Sie können Konfigurationsdateien vorausgesetzte Basiseigenschaften aus den vordefinierten Kategorien für vorausgesetzte Eigenschaften hinzufügen. Alternativ können Sie angepasste vorausgesetzte Eigenschaften hinzufügen.

Informationen zu diesem Vorgang

Wichtig: Sie müssen die Formatierungsregeln für das Hinzufügen und Bearbeiten vorausgesetzter Eigenschaften in einer Konfigurationsdatei einhalten. Wenn Sie dies nicht tun, kann Prerequisite Scanner keinen erfolgreichen Scan für diese vorausgesetzte Eigenschaft durchführen.

Vorgehensweise

1. Öffnen Sie die Konfigurationsdatei.

2. Überprüfen Sie die vorausgesetzten Basiseigenschaften, die in Anhang C, „Referenzinformationen zu vorausgesetzten Eigenschaften", auf Seite 91 beschrieben sind, und bestimmen Sie, welche vorausgesetzten Eigenschaften Sie überprüfen möchten.
3. Geben Sie für jede vorausgesetzte Eigenschaft, die Sie hinzufügen möchten, das Name/Wert-Paar mit den ggf. erforderlichen optionalen Qualifikationsmerkmalen ein.

   Wenn Sie beispielsweise vorausgesetzte Eigenschaften aus der allgemeinen vordefinierten Kategorie hinzufügen möchten, geben Sie nur den Eigenschaftsnamen und den erwarteten Wert ein. Fügen Sie der Datei die folgenden vorausgesetzten Eigenschaften hinzu.

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

   Die vordefinierte Kategorie vorausgesetzter Eigenschaften für das Netz hat beispielsweise die Präfix-ID `network`, und der Name der vorausgesetzten Eigenschaft für die Überprüfung der verfügbaren Ports ist `availablePorts`. Sie können die verfügbaren Ports weiter nach Anwendungssubtypen kategorisieren: `DB2` für einen DB2-Datenbankserver, `WAS` für WebSphere Application Server, `FTP` für das Protokoll FTP. Fügen Sie der Datei die folgenden vorausgesetzten Eigenschaften hinzu.

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

   Die vordefinierte Kategorie vorausgesetzter Eigenschaften für das Betriebssystem hat beispielsweise die Präfix-ID `os`, und der Name der vorausgesetzten Eigenschaft für die Überprüfung des verfügbaren Plattenspeicherplatzes in Dateisystemen ist `space`. Sie können die Überprüfung weiter nach Dateisystemsubtypen kategorisieren: `usr` und `home`. Sie können Werte für die Qualifikationsmerkmale `dir` und `unit` angeben.

   Fügen Sie der Datei die folgenden vorausgesetzten Eigenschaften hinzu.

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

   Wichtig: Sie können die vordefinierten Qualifikationsmerkmale nur mit bestimmten vordefinierten Eigenschaften verwenden. Sehen Sie sich dazu die Beschreibung in Tabelle 5 auf Seite 10 an.
4. Wenn eine vorausgesetzte Eigenschaft nicht in den vordefinierten Kategorien vorausgesetzter Eigenschaften vorhanden ist, fügen Sie das Name/Wert-Paar mit einem optionalen Qualifikationsmerkmal für die angepasste vorausgesetzte Eigenschaft und den Wert hinzu. Stellen Sie sicher, dass Sie das folgende Format verwenden, und geben Sie jede vorausgesetzte Eigenschaft in jeweils einer Zeile an.

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

   Erläuterungen:

   - *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
   - *property_name* ist der Name der vorausgesetzten Eigenschaft.
   - *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
   - *qualifier_name* ist ein optionales Attribut für die vorausgesetzte Eigenschaft. IBM Prerequisite Scanner verwendet dieses Attribut, um die vorausgesetzte

Eigenschaft bzw. den Typ der für die vorausgesetzte Eigenschaft durchzuführenden Prüfung zu qualifizieren. Weitere Informationen finden Sie im Abschnitt „Vordefinierte Qualifikationsmerkmale für vorausgesetzte Eigenschaften" auf Seite 9.

Anmerkung: Sie können mehrere Qualifikationsmerkmale durch Kommas getrennt angeben. Die Gruppe der Qualifikationsmerkmale muss in eckige Klammern (`[]`) eingeschlossen werden.

- *qualifier_value* ist der Wert für das optionale Attribut. Jedes Qualifikationsmerkmal und dessen Wert muss durch einen Doppelpunkt (`:`) begrenzt werden.
- *property_value* ist der Wert für die vorausgesetzte Eigenschaft und kann eine Zeichenfolge (String) oder eine ganze Zahl (Integer) sein.

`env.tcrhome` ist beispielsweise eine angepasste vorausgesetzte Eigenschaft, die in dem mit einer Umgebungsvariablen angegebenen Ausgangsverzeichnis nach Tivoli Common Reporting sucht, und der erwartete Wert muss `True` sein:

```
env.tcrhome=True
```

`env.path.jar` ist eine angepasste vorausgesetzte Eigenschaft, die prüft, ob die JRE in der Umgebungsvariablen `PATH` gesetzt ist, und der erwartete Wert muss `False` sein:

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

Anmerkung: Anschließend müssen Sie die folgenden Dateien erstellen, um die angepasste vorausgesetzte Eigenschaft zu suchen und bei Bedarf zu vergleichen: einen angepassten Collector für die Erfassung des tatsächlichen Werts für die angepasste Eigenschaft und ein angepasstes Auswertungsprogramm nur dann, wenn die Standardvergleichsfunktionen für den Vergleich des tatsächlichen Werts mit dem erwarteten Wert nicht funktionieren.

Vorausgesetzte Eigenschaften bearbeiten

Sie können vorausgesetzte Eigenschaften bearbeiten, die erwarteten Werte für diese vorausgesetzten Eigenschaften ändern oder die zugehörigen Werte der Qualifikationsmerkmale ändern.

Vorbereitende Schritte

Prüfen Sie, ob der neue Wert ein gültiger Wert ist, der von der vorausgesetzten Eigenschaft unterstützt wird. Angenommen, die vorausgesetzte Eigenschaft `Disk` erwartet ein numerisches Format mit der Einheit MB oder GB. Wenn Sie den verfügbaren Plattenspeicherplatz in Terabyte (TB) überprüfen möchten, müssen Sie die Vergleichs-API für TP-Vergleiche erweitern. Außerdem müssen Sie die vorausgesetzte Eigenschaft `Disk` in den entsprechenden Konfigurationsdateien bearbeiten.

Überprüfen Sie die vordefinierten Qualifikationsmerkmale und die gültigen Werte für die vorausgesetzte Eigenschaft, wie im Abschnitt „Vordefinierte Qualifikationsmerkmale für vorausgesetzte Eigenschaften" auf Seite 9 beschrieben.

Vorgehensweise

1. Öffnen Sie die Konfigurationsdatei.
2. Geben Sie für jede vorausgesetzte Eigenschaft, die Sie bearbeiten möchten, den neuen erwarteten Wert ein, oder ändern Sie den Wert für das Qualifikationsmerkmal. Angenommen, ein neuer Systemadministrator ist der Rootbenutzer.

In diesem Fall muss der Wert der vorausgesetzten Eigenschaft `user.userID` geändert werden. Ändern Sie den Wert in den neuen Namen:

```
user.userID=smithj
```

Angenommen, das Qualifikationsmerkmal `type` für die vorausgesetzte Eigenschaft `os.ulimit` hat momentan den Wert `filedescriptorlimit` für die Überprüfung des Grenzwerts für Dateideskriptoren. Jetzt möchten Sie aber einen anderen Grenzwert überprüfen, wie z. B. die Stapelgröße. In diesem Fall ändern Sie den Wert des folgenden Qualifikationsmerkmals für die vorausgesetzte Eigenschaft von

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

in

```
os.ulimit=[type:stacksizelimit]512+,unlimited
```

Wichtig: Sie können die vordefinierten Qualifikationsmerkmale nur mit bestimmten vordefinierten Eigenschaften verwenden. Sehen Sie sich dazu die Beschreibung in Tabelle 5 auf Seite 10 an.

Angepasste Collector für Windows-Systeme erstellen

Sie können angepasste Collector erstellen, wenn Collector aus der Basisgruppe keine Werte für die vorausgesetzten Eigenschaften erfassen, die für das zu installierende Produkt erforderlich sind. Sie können angepasste allgemeine VBScript-Collector erstellen, um Daten für vorausgesetzte Eigenschaften zu erfassen, die für alle Produkte und Produktversionen gelten. Alternativ können Sie angepasste produktspezifische Collector erstellen, um Daten zu erfassen, die nur für ein bestimmtes Produkt und eine bestimmte Produktversion gelten. Obwohl alle angepassten VBScript-Collector Daten mit denselben Methoden erfassen, sind die Regeln für die Erstellung, das Speichern und die Ausführung für jeden Collector geringfügig anders.

Angepasste allgemeine VBScript-Collector für alle Konfigurationsdateien erstellen

Wenn Sie angepasste allgemeine VBScript-Collector erstellen, muss der Dateiname den Namen der vorausgesetzten Eigenschaft enthalten. Diese Collector müssen im Unterverzeichnis `/lib` gespeichert werden. Der Collector enthält den tatsächlichen Wert für eine erforderliche Eigenschaft. Er kann für den Abruf dieser Werte ggf. auch die allgemeinen Funktionen und Subroutinen verwenden.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie sich die Gruppe der vordefinierten Funktionen und Subroutinen in den folgenden Anhängen angesehen haben, bevor Sie die Collector erstellen. Stellen Sie fest, ob Sie eine dieser Funktionen für den Abruf der tatsächlichen Werte verwenden können:

- Anhang E, „Allgemeine Funktionen für Windows-Systeme", auf Seite 123
- Anhang G, „Subroutinen des Dateidienstprogramms auf Windows-Systemen", auf Seite 139
- Anhang F, „Subroutinen des Protokolldienstprogramms auf Windows-Systemen", auf Seite 137
- Anhang H, „Weitere allgemeine Funktionen und Subroutinen für Windows-Systeme", auf Seite 141

Stellen Sie fest, ob der Collector überprüfen muss, ob die vorausgesetzte Eigenschaft vorhanden ist, und wenn ja, welche weiteren Informationen erfasst werden müssen. Jede Prüfung muss einen Wert zurückgeben, der anzeigt, ob die Eigenschaft vorhanden ist oder nicht. Beispiel:

- Prüfen, ob eine Umgebungsvariable vorhanden ist, wie z. B. das Ausgangsverzeichnis eines Produkts, z. B. TCR_HOME für Tivoli Common Reporting.
- Prüfen, ob die Umgebungsvariable eine JAR-Datei, eine Binärdatei oder einen Pfad zur JRE in der Umgebungsvariablen PATH enthält.
- Tatsächlichen Wert einer Umgebungsvariablen prüfen, z. B. das Ausgangsverzeichnis eines Produkts wie TCR_HOME für Tivoli Common Reporting.
- Prüfen, ob ein Produkt installiert ist.
- Prüfen, welche Version des Produkts installiert ist.

Vorgehensweise

1. Erstellen Sie eine VBScript-Datei. Speichern Sie die Datei im Verzeichnis *ips_root*/lib mit einer Variante der folgenden Dateinamenskonvention:

   [*prefix_identifier*.]property_name.vbs

   Erläuterungen:

   - *prefix_identifier* ist die Präfix-ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4.
   - *property_name* ist der Name der vorausgesetzten Eigenschaft und wird im Collectornamen verwendet.

   mssqlVersion.vbs enthält beispielsweise den Code für den Abruf des tatsächlichen Werts für die vorausgesetzte Eigenschaft "MS SQL Server" auf der Windows-Maschine.
2. Fügen Sie mit einem VBScript-Editor den Code hinzu, um den Wert für die vorausgesetzte Eigenschaft abzurufen. Verwenden Sie VBScript COM und VBScript-Funktionen, um auf Elemente der Windows-Umgebung zuzugreifen und sie in der Windows-Script-Host-Umgebung auszuführen. Stellen Sie sicher, dass die Prüfung eine Standardausgabe wie die folgende zurückgibt:

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

   - *property_name* stellt die vorausgesetzte Eigenschaft dar, die in der Konfigurationsdatei angegeben ist, z. B. env.tcrhome.
   - *var_for_value*, d. h. die VBScript-Variable für den tatsächlichen Wert, den der Collector für die vorausgesetzte Eigenschaft abruft.

   Verwenden Sie den folgenden Code, um zu überprüfen, ob die TCR_HOME-Umgebung vorhanden ist, und den tatsächlichen Wert für die vorausgesetzte Eigenschaft env.tcrhome zurückzugeben:

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

   Verwenden Sie den folgenden Code, um zu überprüfen, ob die JRE in der Variablen PATH gesetzt ist, wobei die vorausgesetzte Eigenschaft den Namen env.path.jre hat:

   ```
   Set wshShell = WScript.CreateObject("WScript.Shell")
   path = WshShell.ExpandEnvironmentStrings("%PATH%")
   Set objRegEx = new RegExp
   objRegEx.Pattern = "(^|([:;\\/]))(C:\Program Files\IBM\Java60\jre\bin)($|[:;])"
   objRegEx.IgnoreCase = True
   ```

```
objRegEx.Global = True
Set matches = objRegEx.Execute(path)
WScript.Echo "env.path.jre=" &#38; (matches.Count > 0)
```

Verwenden Sie den folgenden Code, um die Version des installierten Produkts Tivoli Directory Integrator zu überprüfen, wobei die vorausgesetzte Eigenschaft den Namen `installedSoftware.TDI.version` hat:

```
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. Führen Sie den VBScript-Collector aus, um sicherzustellen, dass keine Laufzeitfehler aufgetreten sind, und beheben Sie ggf. aufgetretene Fehler.
4. Erstellen Sie ein angepasstes Auswertungsprogramm nur dann, wenn die tatsächlichen und erwarteten Werte nicht mit den Standardvergleichsfunktionen verglichen werden können.

Spezielle angepasste VBScript-Collector für ein Produkt und eine Produktversion erstellen

Wenn Sie angepasste produktspezifische VBScript-Collector erstellen, muss der Dateiname dem Produktcode in der Konfigurationsdatei entsprechen, und die Collector müssen im Unterverzeichnis `/Windows` gespeichert werden. Der Collector kann Code für die Erfassung der tatsächlichen Werte für eine oder mehrere vorausgesetzte Eigenschaften enthalten. Er kann für die Erfassung dieser Werte ggf. auch die allgemeinen Funktionen und Subroutinen verwenden.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie sich die Gruppe der Funktionen und Subroutinen in den folgenden Anhängen angesehen haben, bevor Sie die Collector erstellen. Stellen Sie fest, ob Sie eine dieser Funktionen für den Abruf der tatsächlichen Werte verwenden können:

- Anhang E, „Allgemeine Funktionen für Windows-Systeme", auf Seite 123
- Anhang G, „Subroutinen des Dateidienstprogramms auf Windows-Systemen", auf Seite 139
- Anhang F, „Subroutinen des Protokolldienstprogramms auf Windows-Systemen", auf Seite 137
- Anhang H, „Weitere allgemeine Funktionen und Subroutinen für Windows-Systeme", auf Seite 141

Stellen Sie fest, ob der Collector überprüfen muss, ob die vorausgesetzte Eigenschaft vorhanden ist, und wenn ja, welche weiteren Informationen erfasst werden müssen. Jede Prüfung muss einen Wert zurückgeben, der anzeigt, ob die Eigenschaft vorhanden ist. Beispiel:

- Prüfen, ob das Verzeichnis vorhanden ist.
- Verfügbaren Plattenspeicherplatz für ein Verzeichnis überprüfen.
- Prüfen, ob ein Produkt installiert ist.
- Prüfen, welche Version des Produkts installiert ist.

Vorgehensweise

1. Erstellen Sie eine VBScript-Datei. Speichern Sie die Datei im Verzeichnis *ips_root*/Windows mit einer Variante der folgenden Dateinamenskonvention:

   product_code[*_version*].vbs

   Erläuterungen:

   - *product_code*

     Dies ist die Variable für die Darstellung eines Produktcodes auf Windows- oder UNIX-Systemen. Produktcodes identifizieren das Produkt, eine individuelle Plattform wie Windows, AIX, HP-UX, Linux oder Solaris und optional die Version des Betriebssystems, das von diesem Produkt unterstützt wird. Sie werden in der Datei codename.cfg gespeichert. Jedes Produkt, das mehrere Plattformen unterstützt, hat mehrere Produktcodes, die jeweils ein Produkt, eine Plattform und eine Version des Betriebssystems identifizieren.
   - *version* ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100.

2. Öffnen Sie die Datei in einem VBScript-Editor, und schließen Sie wie folgt den Pfad zu common_function.vbs ein, wenn Sie allgemeine Funktionen verwenden müssen:

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

3. Wenn Sie die Werte der Umgebungsvariablen PATH verwenden müssen und das Flag -p von Prerequisite Scanner übergeben wird, verwenden Sie Wscript.Arguments(), wobei Wscript.Arguments(0) für den Wert von PATH steht. Wscript.Arguments(1) steht für das Flag -p und dessen Wert.

4. Fügen Sie den Code zum Abrufen des Werts für die vorausgesetzte Eigenschaft mit VBScript COM und VBScript-Funktionen hinzu, um auf die Elemente der Windows-Umgebung zuzugreifen. Führen Sie den Collector in der Windows-Script-Host-Umgebung aus. Stellen Sie sicher, dass die Prüfung eine Standardausgabe wie die folgende zurückgibt:

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

   - *property_name* stellt die vorausgesetzte Eigenschaft dar, die in der Konfigurationsdatei angegeben ist, z. B. env.tcrhome.
   - *var_for_value*, d. h. die VBScript-Variable für den tatsächlichen Wert, den der Collector für die vorausgesetzte Eigenschaft abruft.

   Sie können beispielsweise den verfügbaren Plattenspeicherplatz für das Installationsverzeichnis eines Produkts überprüfen. Verwenden Sie beispielsweise den folgenden Code, wenn Sie Tivoli Monitoring for Energy Management Reporting and Optimization mit der Subroutine „getValue()” auf Seite 142 und der vorausgesetzten Eigenschaft InstallDir überprüfen möchten:

   ```
   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. Erstellen Sie eine Stapeldatei für den Aufruf des VBScript-Collectors. Die Stapeldatei muss denselben Namen wie die Konfigurationsdatei und die Erweiterung `.bat` haben, *product_code*[*_version*]`.bat`. Beispiel:

```
@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. Führen Sie den VBScript-Collector aus, um sicherzustellen, dass keine Laufzeitfehler aufgetreten sind, und beheben Sie ggf. aufgetretene Fehler.
7. Erstellen Sie ein angepasstes Auswertungsprogramm nur dann, wenn die tatsächlichen und erwarteten Werte nicht mit den Standardvergleichsfunktionen verglichen werden können.

Angepasste Collector für UNIX-Systeme erstellen

Sie können angepasste Collector erstellen, wenn Collector aus der Basisgruppe keine Werte für die vorausgesetzten Eigenschaften erfassen, die für das zu installierende Produkt erforderlich sind. Wenn Sie angepasste Collector erstellen, muss der Dateiname der vorausgesetzten Eigenschaft ohne den Subtyp entsprechen. Der Collector wird im Unterverzeichnis `/UNIX_Linux` gespeichert. Der Collector kann Code für den Abruf der tatsächlichen Werte für eine oder mehrere vorausgesetzte Eigenschaften enthalten. Er kann für den Abruf dieser Werte ggf. auch die allgemeinen Funktionen verwenden.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie sich die Gruppe der Funktionen in den folgenden Anhängen angesehen haben, bevor Sie die Collector erstellen. Stellen Sie fest, ob Sie eine dieser Funktionen für den Abruf der tatsächlichen Werte verwenden können:

- Anhang I, „Allgemeine Funktionen für UNIX-Systeme", auf Seite 145
- Anhang J, „Weitere Funktionen für UNIX-Systeme", auf Seite 153
- Anhang K, „Protokolldienstprogrammfunktionen für UNIX-Systeme", auf Seite 161

Stellen Sie fest, ob der Collector überprüfen muss, ob die vorausgesetzte Eigenschaft vorhanden ist, und wenn ja, welche weiteren Informationen erfasst werden müssen. Jede Prüfung muss einen Wert zurückgeben, der anzeigt, ob die Eigenschaft vorhanden ist. Beispiel:

- Prüfen, ob ein Produkt installiert ist, z. B. ein mit RPM installiertes Paket.
- Prüfen, welche Version des Produkts installiert ist.
- Verfügbaren Plattenspeicherplatz für ein angehängtes Dateisystem prüfen.

Wenn Sie Subtypen (*suffix_identifier*) verwenden und eine vorausgesetzte Eigenschaft nach Anwendung, Dienstprogramm oder Servicesubtyp näher kategorisieren möchten, können Sie einen allgemeinen Collector erstellen. Übergeben Sie das Differenzierungsmerkmal für den Subtyp *suffix_identifier*, d. h., *differentiator_suffix_identifier*, an den Collector. `os.package` ist beispielsweise der allgemeine Collector, der die Existenz von Paketen prüft. Wenn Sie die Existenz von `openssh` überprüfen möchten, übergeben Sie wie folgt den Namen des Pakets beim Aufruf des Collectors `os.package` in der Scriptdatei `packageTest.sh`:

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

`openssh` ist hier der Name des Pakets, d. h. Subtyp *suffix_identifier* und Differenzierungsmerkmal *differentiator_suffix_identifier*.

Vorgehensweise

1. Erstellen Sie eine Shell-Script-Datei. Speichern Sie die Datei im Verzeichnis *ips_root*`/Unix_Linux` mit einer Variante der folgenden Dateinamenskonvention, aber ohne Dateierweiterung:

   [*prefix_identifier*.]*property_name*

   Erläuterungen:

   - *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich, zum Beispiel `env`.
   - *property_name* ist der Name der vorausgesetzten Eigenschaft, z. B. `path.jre`.
2. Öffnen Sie die Datei in einem Editor, und schließen Sie wie folgt den Pfad zu `common_function.sh` ein, wenn Sie allgemeine Funktionen verwenden müssen:

   ```
   . ../lib/common_function.sh
   ```
3. Fügen Sie den Code für den Abruf des Werts für die vorausgesetzte Eigenschaft mit den speziellen Befehlen und Optionen für diese Plattform hinzu, um auf Elemente der Hostumgebung zuzugreifen. Die angepasste vorausgesetzte Eigenschaft `env.path.jar` muss beispielsweise überprüfen, ob die JRE in der Variablen `PATH` gesetzt ist. Der folgende Code führt den Befehl `env` aus, durchsucht die Ausgabe für die Variable `PATH` und sucht anschließend den Wert für den JRE-Pfad.

   ```
   envJRE=`env | grep "PATH" | grep -w "/opt/IBM/Java60/jre/bin"`
   ```
4. Stellen Sie sicher, dass die Prüfung eine Standardausgabe zurückgibt:

   ```
   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
   ```

   In diesem Beispiel gibt die Prüfung basierend auf dem Wert der Variablen `$envJRE` entweder `True` oder `False` zurück:

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

5. Führen Sie den angepassten Collector aus, um sicherzustellen, dass keine Laufzeitfehler aufgetreten sind, und beheben Sie ggf. aufgetretene Fehler.
6. Bearbeiten Sie das Script *ips_root*/UNIX_Linux/packageTest.sh für den Aufruf und die Ausführung des angepassten Collectors.
7. Erstellen Sie ein angepasstes Auswertungsprogramm nur dann, wenn der angepasste Collector andere Werte als boolesche Werte zurückgibt.

Pakettestscript für UNIX-Systeme bearbeiten

Sie können die Scriptdatei packageTest.sh aktualisieren, um angepasste Collector auf UNIX-Systemen aufzurufen.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie die Namen der Collector kennen, die den vorausgesetzten Eigenschaften zugeordnet sind. Weitere Informationen finden Sie in Anhang D, „Vordefinierte Collector für UNIX-Systeme", auf Seite 117. Wenn die vorausgesetzte Eigenschaft näher nach Anwendung-, Dienstprogramm- oder Servicesubtyp kategorisiert ist, übergeben Sie das Differenzierungsmerkmal für den Subtyp *suffix_identifier*, d. h., *differentiator_suffix_identifier*, an den Collector.

os.package ist beispielsweise der allgemeine Collector, der die Existenz von Paketen prüft. Wenn Sie die Existenz von openssh überprüfen möchten, übergeben Sie wie folgt den Namen des Pakets beim Aufruf des Collectors os.package in der Scriptdatei packageTest.sh:

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

openssh ist der Name des Pakets, d. h. der Subtyp *suffix_identifier*, und *differentiator_suffix_identifier* ist das Differenzierungsmerkmal.

Vorgehensweise

1. Öffnen Sie in einem Editors das Script *ips_root*/UNIX_Linux/packageTest.sh.
2. Fügen Sie den Code hinzu, der die angepasste vorausgesetzte Eigenschaft aus der Konfigurationsdatei liest und deren Wert parst.

   ```
   res=`echo $line | grep [prefix_identifier.]property_name[.suffix_identifier]`
   if [ $res ]; then
   ExpValue=`echo $res | cut -d "=" -f2`
   ```

   Verwenden Sie beispielsweise den folgenden Code, um die angepasste vorausgesetzte Eigenschaft env.path.jar einzulesen und zu prüfen, ob die JRE in der Variablen PATH gesetzt ist:

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

   Beispiel:

   ```
    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. Rufen Sie die Protokollierungsfunktionen für Trace- und Debugdaten auf, bevor Sie den angepassten Collector aurufen.

```
 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. Rufen Sie den angepassten Collector auf.

   Anmerkung: Wenn der angepasste Collector Subtypen, d. h. [*suffix_identifier*] im Dateinamen enthält und weitere Prüfungen basierend auf dem Subtyp erfordert, übergeben Sie das Differenzierungsmerkmal [*differentiator_suffix_identifier*] für den Subtyp an den angepassten Collector:

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

   Beispiel:

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

   Anmerkung: Beispiele für Differenzierungsmerkmale für den Subtyp *script_name* für die vorausgesetzte Eigenschaft `os.file.script_name` sind die Pfade zu den Scripts, die an den Collector `os.filepath` übergeben werden:

   ```
   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. Rufen Sie die Protokollierungsfunktionen für Trace- und Debugdaten beim Beenden des angepassten Collectors auf:

   ```
    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
   ```

   Beispiel:

   ```
    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. Wiederholen Sie die Schritte 2 bis 5 für jede angepasste vorausgesetzte Eigenschaft.

Angepasste Auswertungsprogramme für Windows-Systeme erstellen

Sie können VBScript-Auswertungsprogramme erstellen, wenn die Basisauswertungsprogramme die erwarteten und die tatsächlichen Werte der vorausgesetzten Eigenschaften nicht mit den richtigen Bewertungskriterien vergleichen. Wenn Sie angepasste Auswertungsprogramme erstellen, muss die Datei die Dateierweiterung `_compare` haben und im Unterverzeichnis `/Windows` gespeichert werden. Das ange-

passte Auswertungsprogramm kann die allgemeinen Funktionen und Subroutinen verwenden, um ggf. die Werte zu vergleichen.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie sich die Gruppe der Funktionen und Subroutinen in den folgenden Anhängen angesehen haben, bevor Sie das Auswertungsprogramm erstellen. Stellen Sie fest, ob Sie eine dieser Funktionen und Subroutinen für den Vergleich der Werte verwenden können:

- Anhang E, „Allgemeine Funktionen für Windows-Systeme", auf Seite 123
- Anhang G, „Subroutinen des Dateidienstprogramms auf Windows-Systemen", auf Seite 139
- Anhang F, „Subroutinen des Protokolldienstprogramms auf Windows-Systemen", auf Seite 137
- Anhang H, „Weitere allgemeine Funktionen und Subroutinen für Windows-Systeme", auf Seite 141

Anmerkung: Die allgemeine Funktion „passOrFail()" auf Seite 134 kann die tatsächlichen und die erwarteten Werte für die folgenden Datentypen vergleichen: eine generische Nummer, Größe in MB oder GB, Prozessorgeschwindigkeit in MHz oder GHz, boolescher Wert oder Zeichenfolge. Erstellen Sie nur dann ein angepasstes Auswertungsprogramm, wenn die Funktion `passOrFail` nicht verwendet werden kann.

Vorgehensweise

1. Erstellen Sie eine VBScript-Datei. Speichern Sie die Datei im Verzeichnis *ips_root*`/Windows` mit einer Variante der folgenden Dateinamenskonvention:

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

   Erläuterungen:

   - *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
   - *property_name* ist der Name der vorausgesetzten Eigenschaft.
   - *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
2. Fügen Sie den Code hinzu, um die tatsächlichen und erwarteten Werte, die mit VBScript COM als Argumente an das Auswertungsprogramm und an die zugehörigen Funktionen übergeben werden, zu vergleichen. Stellen Sie sicher, dass der Vergleich eine Standardausgabe wie die folgende zurückgibt:
   - `"PASS"`, wenn der erwartete Wert für die vorausgesetzte Eigenschaft größergleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist
   - `"FAIL"`, wenn der erwartete Werte für die vorausgesetzte Eigenschaft nicht gleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist
3. Führen Sie das angepasste Auswertungsprogramm aus, um sicherzustellen, dass keine Laufzeitfehler aufgetreten sind, und beheben Sie ggf. aufgetretene Fehler.

Beispiel

Dieses angepasste Auswertungsprogramm überprüft die tatsächlichen und erwarteten Werte für die Version von Tivoli Directory Integrator. Es verwendet die allgemeine Funktion „versionCompare()" auf Seite 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
        ' Version must be at least expected value
        if (cmp=0 or cmp=-1) Then
            tdiVersionCompare = "PASS"
        else
            tdiVersionCompare = "FAIL"
        end if
    elseif (StrComp(Right(expect,1),"-")=0) Then
            ' Version must be less than or equal to expected value
            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

' Generische Funktion für den Vergleich von zwei Versionszeichenfolgen
'
' Parameter
'       ver1 Die erste Versionszeichenfolge
'       ver2 Die zweite Versionszeichenfolge
'
' Für ver1 und ver2 werden durch Punkte getrennte Versionszeichenfolgen
' erwartet (z. B. 1.0.0.4, 2.3, 3.40.26.7800, 2.3.a). Versionszeichenfolgen
' können eine beliebige Anzahl an Teilen haben Wenn Versionen mit einer
' unterschiedlichen Anzahl an Teilen verglichen werden, werden fehlende
' Teile der kürzeren Versionszeichenfolge so behandelt, als enthielten
' sie eine null. Wenn ein Versionsteil nicht numerische Zeichen enthält,
' werden die entsprechenden Teile als Zeichenfolgen verglichen und
' nicht in das numerische Format geparst.
'
' Rückgabewerte:
'       1 version1 > version2
'      -1 version1 &#60; version2
'       0 version1 = version2
'
' Sonderfälle:
' RESULT    version 1    version 2
```

```
'   0         empty        empty
'   1      validString     empty
'  -1         empty     validString
'
' ANMERKUNG: Diese Funktion sollte letztendlich in common_functions.vbs verschoben werden.

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

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

    ' Sonderfälle behandeln:
    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

    ' Versionen sind nicht leer. Aufteilen und Nummern vergleichen.
    ver1Parts = Split(ver1,".")
    ver2Parts = Split(ver2,".")

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

    ' Wenn der letzte Versionsteil "*" ist, alle Teile als "*" behandeln.
    '(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
            ' Als Zeichenfolgen vergleichen, wenn ein Teil nicht in eine Nummer konvertiert werden ka
            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

    ' Hier müssen die Versionen identisch sein
    versionCompare = 0

end function
```

Angepasste Auswertungsprogramme für UNIX-Systeme erstellen

Sie können angepasste Auswertungsprogramme erstellen, wenn der angepasste Collector keine boolesche Werte zurückgibt, d. h. `True` oder `False`. Wenn Sie angepasste Auswertungsprogramme erstellen, müssen diese die Dateierweiterung `_compare` haben und im Unterverzeichnis `/UNIX_Linux` gespeichert werden. Das angepasste Auswertungsprogramm kann die allgemeinen Funktionen verwenden, um ggf. die Werte zu vergleichen.

Vorbereitende Schritte

Stellen Sie sicher, dass Sie sich die Gruppe der Funktionen in den folgenden Anhängen angesehen haben, bevor Sie die angepassten Auswertungsprogramme erstellen. Stellen Sie fest, ob Sie eine dieser Funktionen für den Vergleich der tatsächlichen und erwarteten Werte verwenden können:

- Anhang I, „Allgemeine Funktionen für UNIX-Systeme", auf Seite 145
- Anhang J, „Weitere Funktionen für UNIX-Systeme", auf Seite 153
- Anhang K, „Protokolldienstprogrammfunktionen für UNIX-Systeme", auf Seite 161

Es gibt zwei Scriptdateien, die Sie als Ausgangspunkt verwenden können, d. h., `._compare.sh` und `_compare.sh`, im Unterverzeichnis `/Unix_Linux`.

Wichtig: Erstellen Sie keine angepassten Auswertungsprogramme, wenn Ihre angepassten Collector `True` oder `False` zurückgeben. IBM Prerequisite Scanner verwendet vordefinierte Auswertungsprogramme für alle Collector, die boolesche Werte zurückgeben.

Vorgehensweise

1. Erstellen Sie eine Shell-Datei. Speichern Sie die Datei im Verzeichnis *ips_root*`/UNIX_Linux` mit einer Variante der folgenden Dateinamenskonvention:

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

   Erläuterungen:

   - *prefix_identifier* ist eine ID für eine vordefinierte Kategorie vorausgesetzter Eigenschaften. Weitere Informationen finden Sie in Tabelle 3 auf Seite 4. Diese Präfix-ID ist für einige der vordefinierten Kategorien erforderlich.
   - *property_name* ist der Name der vorausgesetzten Eigenschaft.
   - *suffix_identifier* ist eine optionale ID für einen Subtyp vorausgesetzter Eigenschaften. Weitere Informationen hierzu finden Sie in Tabelle 4 auf Seite 7.
2. Fügen Sie den Code hinzu, um die tatsächlichen und erwarteten Werte, die als Argumente an das Auswertungsprogramm und die zugehörigen Funktionen übergeben werden, zu vergleichen. Stellen Sie sicher, dass der Vergleich eine Standardausgabe wie die folgende zurückgibt:
   - `"PASS"`, wenn der erwartete Wert für die vorausgesetzte Eigenschaft größergleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist
   - `"FAIL"`, wenn der erwartete Werte für die vorausgesetzte Eigenschaft nicht gleich dem tatsächlichen Wert für die vorausgesetzte Eigenschaft ist
3. Führen Sie das angepasste Auswertungsprogramm aus, um sicherzustellen, dass keine Laufzeitfehler aufgetreten sind, und beheben Sie ggf. aufgetretene Fehler.

Kapitel 4. Prerequisite Scanner ausführen

Sie können eine Befehlszeilenschnittstelle verwenden, um IBM Prerequisite Scanner auszuführen. Das Script `prereq_checker` von Prerequisite Scanner akzeptiert eine Reihe erforderlicher und optionaler Parameter und ein Befehlsflag für weitere optionale Parameter.

In Tabelle 12 sind die Sonderzeichen beschrieben, die in der Syntax des Scripts von Prerequisite Scanner verwendet werden.

Tabelle 12. Legende der Sonderzeichen für das Script von Prerequisite Scanner

Sonderzeichen	Beschreibung
<>	Gibt einen Platzhalternamen an.
[]	Gibt einen optionalen Parameter an. Sie müssen Parameter angeben, die nicht in eckige Klammern eingeschlossen sind.
...	Gibt an, dass Sie mehrere Werte für einen Parameter festlegen können.
|	Gibt sich gegenseitig ausschließende Parameter an. Geben Sie entweder den Parameter links vom Trennzeichen oder den Parameter rechts vom Trennzeichen an, aber nicht beide Parameter.
{}	Schließt eine Gruppe sich gegenseitig ausschließender Parameter ein, die durch | getrennt sind.

prereq_checker

Das Script `prereq_checker` führt IBM Prerequisite Scanner aus und überprüft die Voraussetzungen basierend auf den Parametern, die Sie angeben, wenn Sie das Script ausführen.

Syntax

```
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]
```

Das Script `prereq_checker` hat einen erforderlichen Parameter und mehrere optionale Parameter.

„"Product_Code [Product_Version][,Product_CodeN [Product_VerN]]..."" auf Seite 68
Erforderlicher Parameter

„[detail]" auf Seite 68
Optionaler Parameter

„[outputDir="ips_output_dir"]" auf Seite 71
Optionaler Parameter

„[xmlResult]" auf Seite 71
Optionaler Parameter

„[PATH="product_root"]" auf Seite 71

„[-p Product_Code.Instanz.parameter=value,...]" auf Seite 72
Optionales Flag

„[debug]" auf Seite 72
Optionaler Parameter

„[trace]" auf Seite 72
Optionaler Parameter

"*Product_Code* [*Product_Version*][,*Product_CodeN* [*Product-_VerN*]]..."

Sie müssen mindestens einen Parameter **Product_Code** setzen, um das Produkt oder die Komponente anzugeben, für das bzw. die die Prüfung der Voraussetzungen durchgeführt werden soll, und die zugehörige Konfigurationsdatei angeben. **Product_Code** ist der Produktcode, den Sie in der Datei *ips_root*/codename.cfg festlegen.

KMS ist beispielsweise der Produktcode für Tivoli Enterprise Monitoring Server in der Datei product.cfg. Zum Ausführen des Scanners geben Sie das folgende Script mit dem Produktcode ein:

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

Wenn Sie einen Parameter **Product_Code** setzen, der keine entsprechende Konfigurationsdatei hat, ignoriert Prerequisite Scanner ihn ohne Fehler. Die Protokolldatei enthält eine Nachricht, die angibt, dass keine Konfigurationsdatei gefunden wurde.

Der Parameter **Product_Version** für den zugehörigen Parameter **Product_Code** gibt die Version des Produkts an. Der Parameter ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100. **Product_Version** ist ein optionaler Parameter. Wenn Sie diesen Parameter nicht setzen, prüft Prerequisite Scanner die neueste verfügbare Version.

Sie können beliebig viele Parameter **Product_Code** mit dem optionalen Parameter **Product_Version**, jeweils durch Kommas getrennt, angeben.

Wichtig: Wenn Sie mehrere Parameter **Product_Code** mit dem optionalen Parameter **Product_Version** angeben, schließen Sie die Parameter in Anführungszeichen ein. Andernfalls schlägt die Ausführung des Scanners fehl.

In diesem Beispiel werden die Voraussetzungen für die neueste Version von Tivoli Monitoring Operating System Agent for Windows und Version 6.2.1 von Tivoli Monitoring Agent for DB2 geprüft.

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

[detail]

Dieser optionale Parameter gibt an, ob detaillierte Ergebnisse des Scans in der Befehlszeilenschnittstelle angezeigt werden sollen.

Wichtig: Schließen Sie diesen Parameter nicht in Anführungszeichen ein.

Wenn Sie den Parameter **detail** setzen, enthalten die detaillierten Ergebnisse folgende Informationen:

- Version von Prerequisite Scanner
- Version des Betriebssystems, unter dem der Scanner ausgeführt wurde
- Namen der Produkte oder Komponenten, für die die Prüfungen der Voraussetzungen ausgeführt wurden
- Für jede vorausgesetzte Eigenschaft der Name der geprüften vorausgesetzten Eigenschaft, das Ergebnis `PASS` oder `FAIL`, der tatsächliche Wert und der erwartete Wert
- Für alle Komponenten der Name der allgemeinen vorausgesetzten Eigenschaft, das Ergebnis `PASS` oder `FAIL`, der tatsächliche Wert und der erwartete Wert
- Gesamtergebnis `PASS` oder `FAIL`

Prerequisite Scanner speichert diese Ergebnisse auch in der Datei *ips_output_dir*`/result.txt`. Die Ergebnisse werden in der Textdatei gespeichert, unabhängig davon, ob Sie den Parameter **detail** setzen oder nicht.

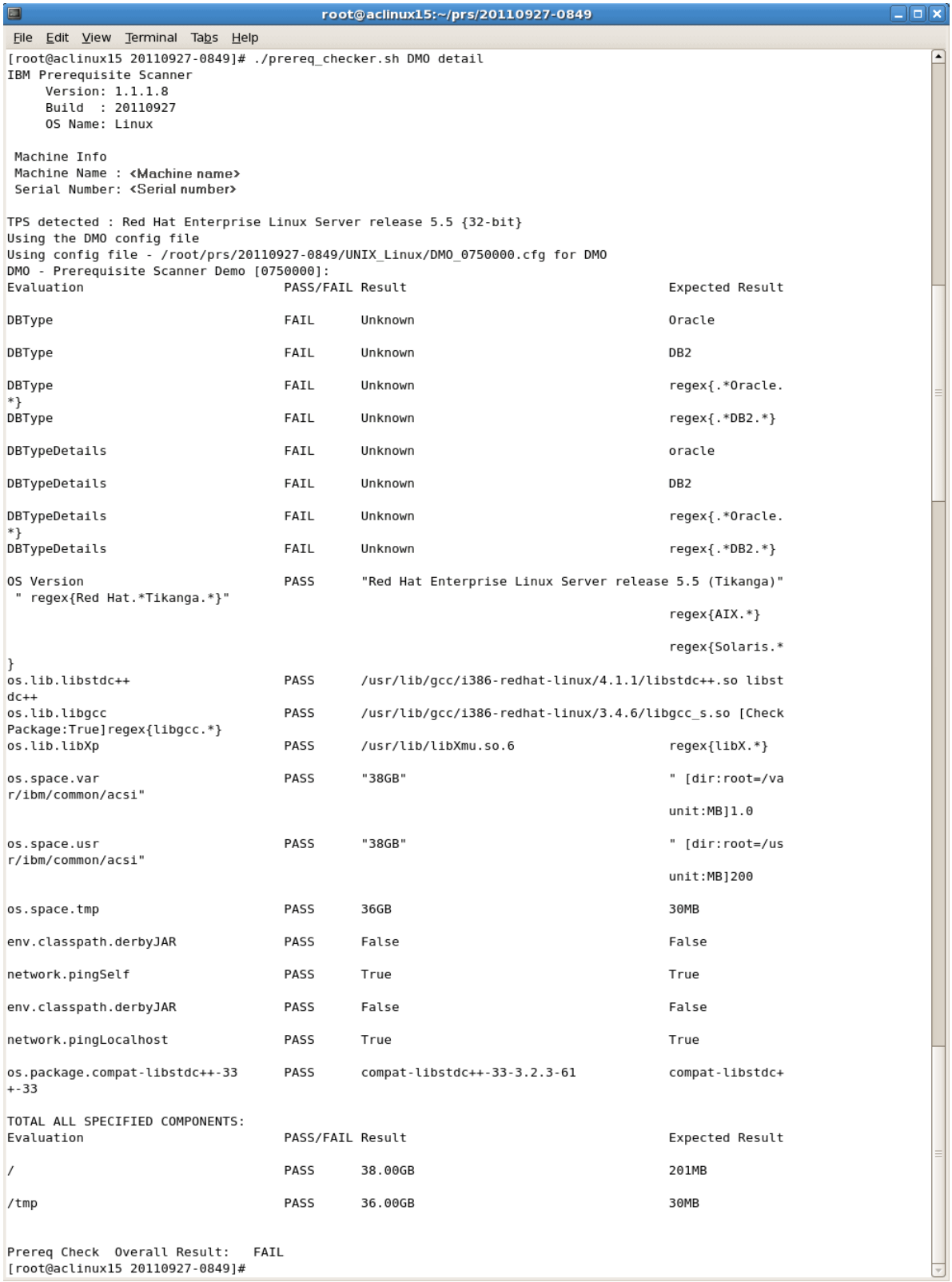


Abbildung 10. Script ausführen und Parameter "detail" setzen - UNIX-Systeme

Wenn Sie den Parameter **detail** nicht setzen, zeigt der Scanner nur das Ergebnis PASS oder FAIL in der Befehlszeilenschnittstelle an.

Abbildung 11. Script ohne den Parameter "detail" ausführen - Windows-Systeme

[outputDir="*ips_output_dir*"]

Dieser optionale Parameter gibt an, dass Sie das Ausgabeverzeichnis für die Scanergebnisse und die Protokolldateien von Prerequisite Scanner definieren möchten.

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **outputDir** setzen, gibt Prerequisite Scanner Ergebnistext-, -XML und Protokolldateien in dem mit dem Parameter angegebenen Verzeichnis aus. Dieser Wert wird in der Dokumentation als *ips_output_dir* bezeichnet.

Wenn Sie diesen Parameter nicht setzen, wird *ips_root* als Standardausgabeposition verwendet.

Sie müssen den Parameter verwenden, um eine Position anzugeben, wenn Sie Prerequisite Scanner von einer CD, einer DVD oder über ein schreibgeschütztes Netzlaufwerk ausführen. Sie müssen Schreibberechtigungen besitzen, um in *ips_output_dir* schreiben zu können. Andernfalls schlägt die Ausführung von Prerequisite Scanner fehl.

Wichtig: Wenn das Ausgabeverzeichnis nicht vorhanden ist, erstellt Prerequisite Scanner das Verzeichnis. Sie müssen Schreibberechtigungen besitzen, um das Ausgabeverzeichnis, in dem Prerequisite Scanner die Dateien speichert, zu erstellen oder zu beschreiben.

[xmlResult]

Dieser optionale Parameter gibt an, dass Sie die Ergebnisse nicht nur in eine einfache Testergebnisdatei, sondern auch in die XML-Ergebnisdatei ausgeben möchten.

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **xmlResult** setzen, gibt Prerequisite Scanner die Ergebnisse in die Datei *ips_output_dir*`/result.xml` aus.

Wenn Sie diesen Parameter nicht setzen, werden die Ergebnisse nur in die einfache Textdatei ausgegeben.

[PATH="*product_root*"]

Dieser optionale Parameter gibt die Installationsverzeichnisse für die Produkte an.

Wichtig: Setzen Sie unter Windows den Pfad nicht auf einen Laufwerkbuchstaben, d. h. `C:`. Stellen Sie sicher, dass Sie einen gültigen Pfad angeben.

Wenn Sie den Parameter **path** nicht setzen, prüft der Scanner die Standardinstallationsverzeichnisse für IBM Tivoli-Produkte:

- Auf **UNIX-Systemen:** `/opt/ibm/itm`

- Auf **Windows-Systemen:** `C:\IBM\itm`

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

Das optionale Flag **-p** gibt an, dass die vorherigen Parameter an eine Scriptdatei übergeben werden müssen, damit weitere Prüfungen der Voraussetzungen ausgeführt werden. **<Product_Code>** ist der Produktcode. Es wird nur eine Gruppe von *instance.parameter=value* an das Script übergeben. Sie können mehrere Gruppen von Parametern durch Kommas getrennt übergeben.

Das Script, an das die Parameter übergeben werden, wird mit den folgenden Optionen bestimmt:

- Mit einem **Product_Code**-Präfix werden die Parameter an das Script mit dem zugehörigen **Product_Code** übergeben.
- Ohne das **Product_Code**-Präfix werden die Parameter an die allgemeinen Collector übergeben.

Beispiel 1`-p KUD.inst1.DB2_INST_OWNER=db2inst1,`
`KUD.inst2.DB2_INST_OWNER=db2inst2` Dieses Flag mit Parametern übergibt `db2inst1.DB2_INST_OWNER=db2inst1` und `db2inst2.DB2_INST_OWNER=db2inst2` an die Scriptdatei `KUD.`**`Product_Version`**`.bat`.

Beispiel 2

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

Dieses Flag mit Parametern übergibt `SERVER=IP.PIPE://mymachine:1918` zur Überprüfung der Ports an den allgemeinen Collector.

Anmerkung: Dieses Script akzeptiert die Parameter in **-p** als `tacmd createNode`. Sie können die Parameter `SERVER`, `PROTOCOL`, `PORT`, `BACKUP` und `BSERVER` in *ips_root*`/lib/common_configuration` setzen. Prerequisite Scanner priorisiert die über die Befehlszeilenschnittstelle übergebenen Parameter vor den Parametern, die in der Datei `common_configuration` definiert sind.

[debug]

Dieser optionale Parameter gibt an, dass Sie das Debugging während der Ausführung von Prerequisite Scanner aktivieren möchten.

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **debug** setzen, gibt Prerequisite Scanner detaillierte Informationen, Warnungen und Fehlernachrichten zur Verarbeitung und die Scanergebnisse in der Protokolldatei aus. Dies ist die Datei *ips_output_dir*`/prs.debug` auf UNIX-Systemen und die Datei *ips_output_dir*`/precheck.log` auf Windows-Systemen.

Wichtig: Das Debugging für den Scanner ist standardmäßig inaktiviert.

[trace]

(nur UNIX-Systeme) Dieser optionale Parameter gibt an, dass Sie die Traceprotokollierung während der Ausführung von Prerequisite Scanner aktivieren möchten.

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **trace** setzen, gibt Prerequisite Scanner Traceinformationen in der Datei *ips_output_dir*`/prs.trc` aus.

Wichtig: Die Traceprotokollierung für den Scanner ist standardmäßig inaktiviert.

Prerequisite Scanner über die Befehlszeile ausführen

Sie können IBM Prerequisite Scanner über die Befehlszeilenschnittstelle ausführen und die entsprechenden Eingabeparameter für das Script eingeben.

Vorbereitende Schritte

Lesen Sie in der Installationsdokumentation zum Produkt oder in den technischen Hinweisen nach, welche zusätzlichen Schritte vor der Ausführung von Prerequisite Scanner ausgeführt werden müssen. Möglicherweise müssen Sie beispielsweise die Umgebungsvariable setzen, die Prerequisite Scanner anzeigt, welche Komponenten oder Features auf dem Zielsystem installiert werden, und damit, welche Voraussetzungen zu prüfen sind.

Vorgehensweise

1. Öffnen Sie die Befehlszeilenschnittstelle und dann das Verzeichnis *ips_root*.
2. Führen Sie die Scriptdatei **prereq_checker** von Prerequisite Scanner wie folgt aus:

   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,...]
   ```

   Das folgende Beispiel führt Prerequisite Scanner für Autonomic Deployment Engine mit einer Konfigurationsdatei und dem zugehörigen Produktcode ADE aus:

   ```
   ./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,...]
   ```

   Das folgende Beispiel führt Prerequisite Scanner für Tivoli Provisioning Manager for Windows 2003 und 2008 mit den Produktcodes COX und COY aus.

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

   Das folgende Beispiel führt Prerequisite Scanner für Tivoli zEnterprise Monitoring Agent mit dem Produktcode KZE aus. Außerdem wird die Position der Ergebnisse und Protokolldateien mit dem optionalen Parameter **outputDir** auf *ips_output_dir* gesetzt.

Wichtig: Sie müssen den Parameter **outputDir** verwenden, um eine Position anzugeben, wenn Sie sich für die Ausführung von Prerequisite Scanner über eine CD, DVD oder ein schreibgeschütztes Netzlaufwerk entscheiden. Sie müssen Schreibberechtigungen besitzen, um in *ips_output_dir* schreiben zu können. Andernfalls schlägt die Ausführung von Prerequisite Scanner fehl.

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

Der Scanner gibt die Dateien `result.txt file` und `precheck.log` an den folgenden Positionen aus:

- Auf Windows-Systemen: `D:\temp\ips`, wobei `TEMP` die Umgebungsvariable für den temporären Ordner ist.
- Auf UNIX-Systemen: `/tmp/ips`

Wichtig: Wenn das Ausgabeverzeichnis nicht vorhanden ist, erstellt Prerequisite Scanner das Verzeichnis. Sie müssen Schreibberechtigungen besitzen, um das Ausgabeverzeichnis, in dem Prerequisite Scanner die Dateien speichert, zu erstellen oder zu beschreiben.

Allgemeine Verzeichnispositionen

Es gibt Pfadnamensvariablen für allgemeine Verzeichnisse.

Installationsverzeichnis von IBM Prerequisite Scanner

ips_root beschreibt die Position, unter der Prerequisite Scanner installiert ist. Diese Position kann während der Installation festgelegt werden.

Ausgabeverzeichnis von Prerequisite Scanner

ips_output_dir beschreibt die Position, an der die Scanergebnisse und Protokolldateien für Prerequisite Scanner gespeichert werden. Diese Position kann mit dem Eingabeparameter **outputDir** bei der Ausführung des Scanners angegeben werden. Wenn Sie diesen Parameter nicht setzen, wird *ips_root* als Standardausgabeposition verwendet.

Anmerkung: Prerequisite Scanner erstellt während der Ausführung temporäre Dateien, jedoch werden diese vor Abschluss der Scannerausführung gelöscht. Diese temporären Dateien befinden sich im Unterverzeichnis *ips_output_dir*`/temp`. Der Scanner löscht auch das Unterverzeichnis *ips_output_dir*`/temp`, sofern das Unterverzeichnis keine Debug- und Tracedateien enthält, die ausschließlich auf UNIX-Systemen generiert wurden.

Kapitel 5. Fehlerbehebung für Prerequisite Scanner

Sie können Fehler in IBM Prerequisite Scanner beheben, indem Sie Protokolldateien und Protokollierungsfunktionen verwenden, wenn Sie angepasste Prüfungen der Voraussetzungen erstellen.

Prerequisite Scanner generiert Rückgabecodes, die von den Ergebnissen des Scans und davon abhängig sind, ob der Scanner aufgrund von Fehlern beendet werden muss. Diese Rückgabecodes werden in die Protokolldateien geschrieben. Wenn Prerequisite Scanner den Scan beispielsweise nicht ausführen kann, weil die Konfigurationsdatei nicht gelesen werden kann, wird der Rückgabecode 2 generiert.

Fehlerbehebung auf Windows-Systemen

Wenn Sie IBM Prerequisite Scanner ausführen, wird standardmäßig eine Protokolldatei erstellt. Diese Datei enthält detaillierte Informationen zu jedem Schritt und jeder Funktion, die der Scanner nacheinander ausführt. Die Datei enthält außerdem die Zeitmarken, einschließlich Start- und Endzeiten, der einzelnen Funktionen und Schritte. Sie können die Protokolldatei debuggen und prüfen, um zu bestimmen, wo und wann der Fehler aufgetreten ist.

Prerequisite Scanner gibt Verarbeitungsinformationen, Warnungen und Fehlernachrichten sowie Scanergebnisse in der Datei *ips_output_dir*`/precheck.log` aus. Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **debug** setzen, gibt Prerequisite Scanner weitere Debugnachrichten in dieser Datei aus.

Abb. 12 auf Seite 76 zeigt eine Beispielprotokolldatei, die erstellt wird, wenn der optionale Parameter **debug** gesetzt ist, und Abb. 13 auf Seite 77 zeigt die Protokolldatei, die erstellt wird, wenn der Parameter nicht gesetzt ist.

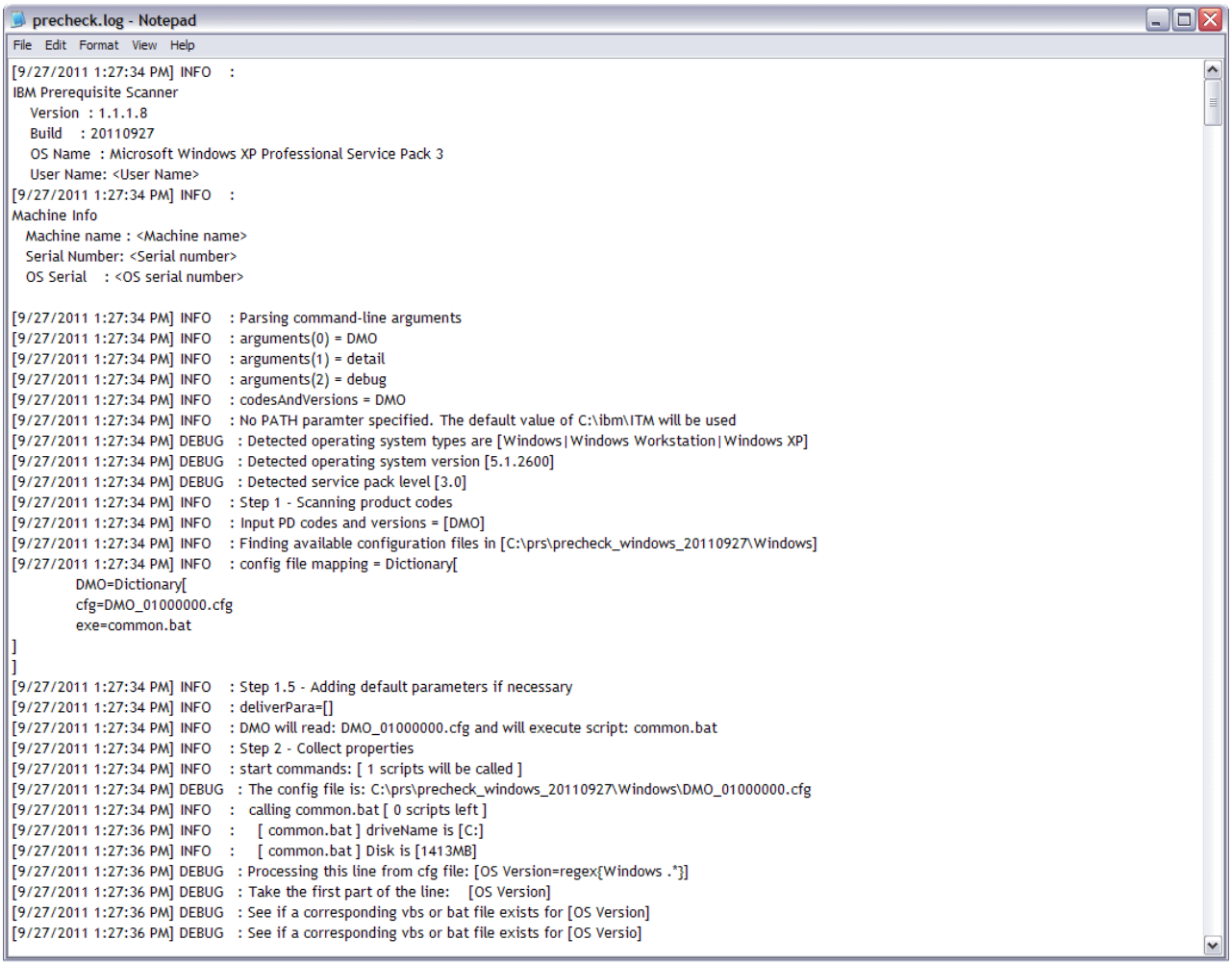


Abbildung 12. Datei "precheck.log" mit Debugdaten

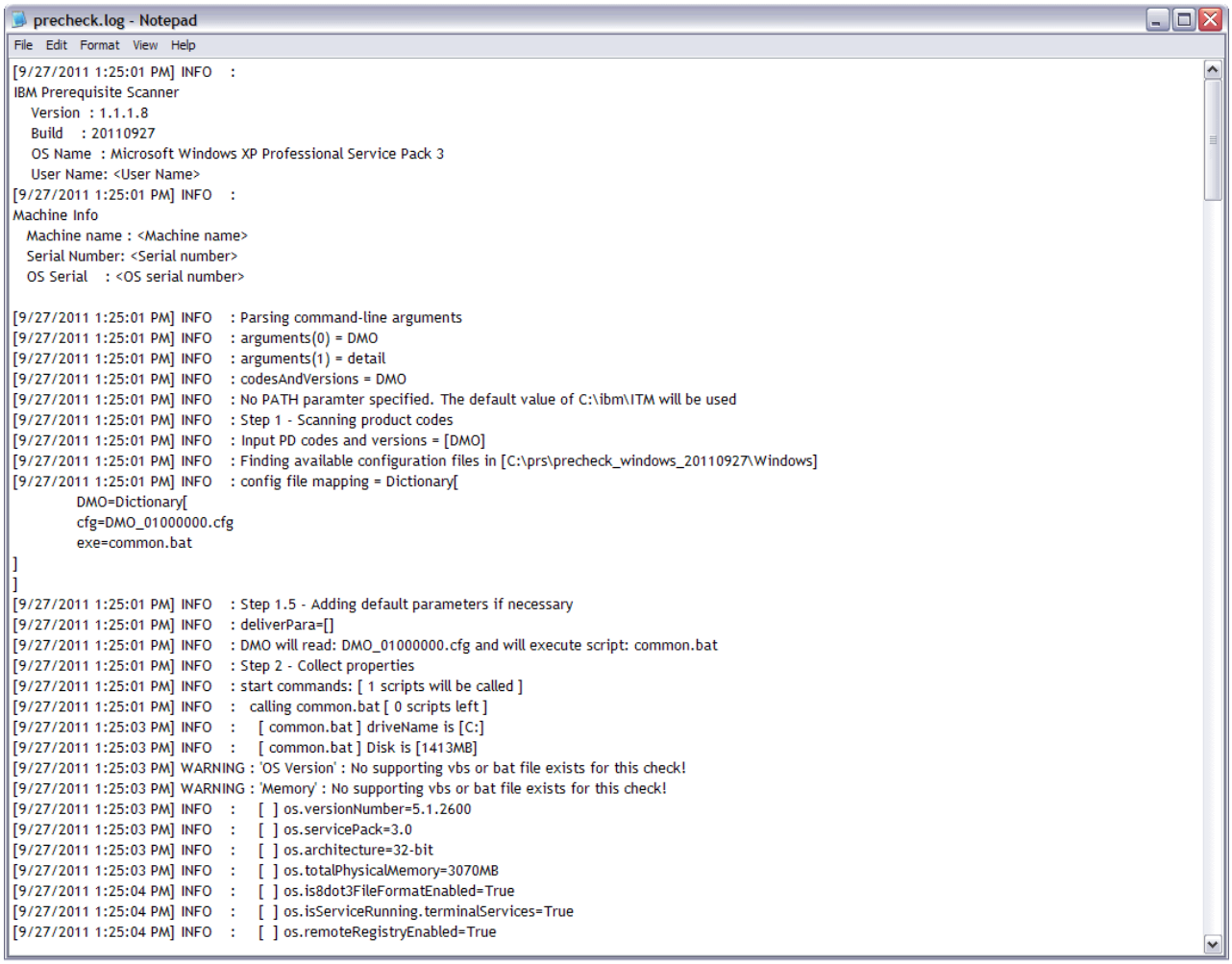


Abbildung 13. Datei "precheck.log" ohne Debugdaten

Fehlerbehebung auf UNIX-Systemen

Das Schreiben von Nachrichten in Protokolldateien ist auf UNIX-Systemen standardmäßig inaktiviert. Sie können die Debug- und Tracefunktionen mit den Eingabeparametern **debug** und **trace** aktivieren. Der Scanner schreibt die Debug-und Tracedaten in verschiedene Protokolldateien und verwendet Zeitmarken, um die Start- und Endzeiten der Schritte oder Funktionen zu kennzeichnen. Sie können beide Dateien verwenden, um ein bestimmtes Problem, eine bestimmte Funktion oder eine bestimmte Prüfung der Voraussetzungen zu korrelieren und die Fehlerbehebung durchzuführen.

Debugprotokolldatei

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **debug** setzen, gibt Prerequisite Scanner detaillierte Informationen, Warnungen und Fehlernachrichten zur Verarbeitung und die Scanergebnisse in der Datei *ips_output_dir*/temp/prs.debug aus. Diese Datei enthält detaillierte Informationen zu jedem Schritt und jeder Funktion, die der Scanner nacheinander ausführt. Die Datei enthält außerdem die Zeitmarken, einschließlich Start- und Endzeiten, der

einzelnen Funktionen und Schritte. Das Unterverzeichnis *ips_output_dir*/temp enthält außerdem die vorläufigen Dateien result1.txt und result2.txt, die die Eingabe für die endgültige Datei *ips_output_dir*/result.txt bereitstellen. Sie können diese vorläufigen Dateien verwenden, um Probleme mit den Ergebnissen für bestimmte Prüfungen der Voraussetzungen zu bestimmen.

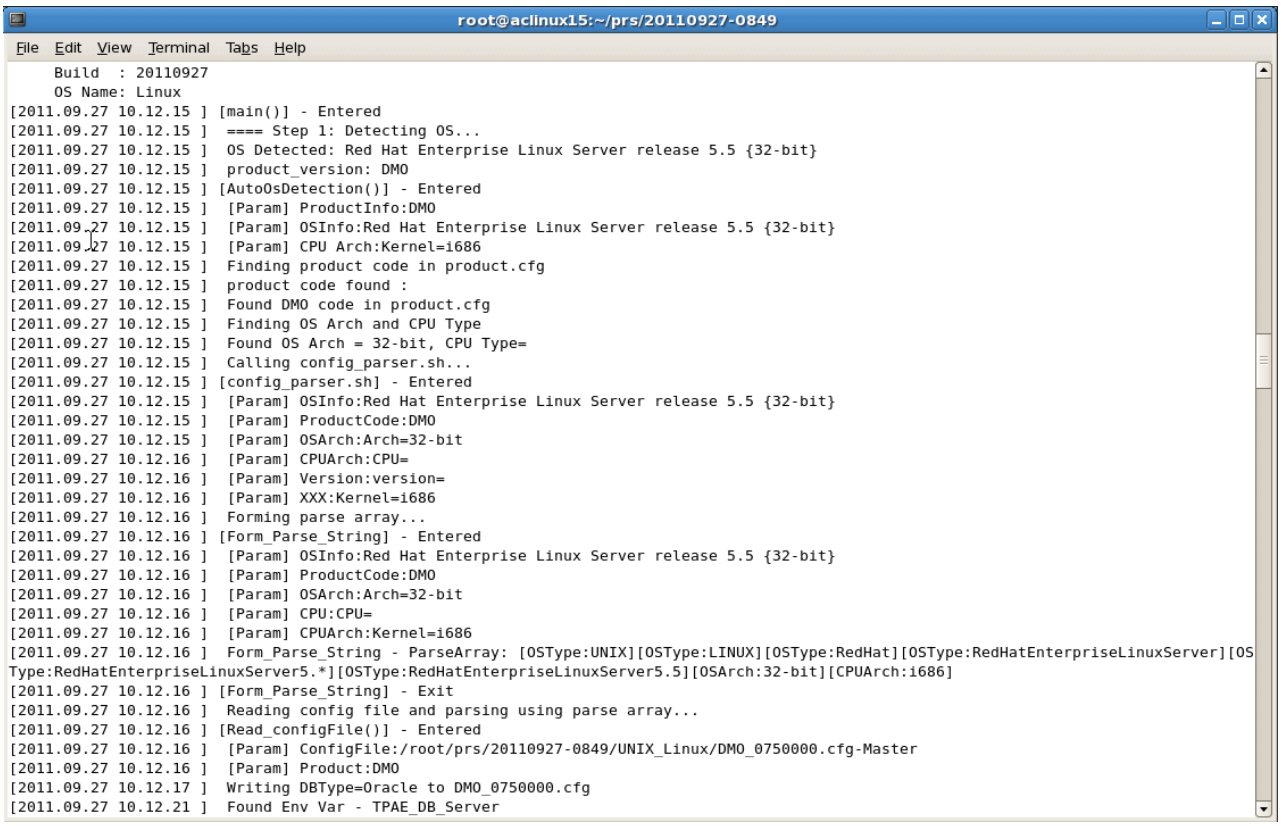


Abbildung 14. Datei "prs.debug" auf UNIX-Systemen

Traceprotokolldatei

Wenn Sie das Script von Prerequisite Scanner ausführen und den optionalen Parameter **trace** setzen, gibt Prerequisite Scanner Traceinformationen in der Datei *ips_output_dir*/temp/prs.trc aus. Diese Datei enthält Informationen zu allen Funktionen, die der Scanner nacheinander ausführt. Die Datei enthält außerdem die Zeitmarken, einschließlich Start- und Endzeiten, der einzelnen Funktionen.

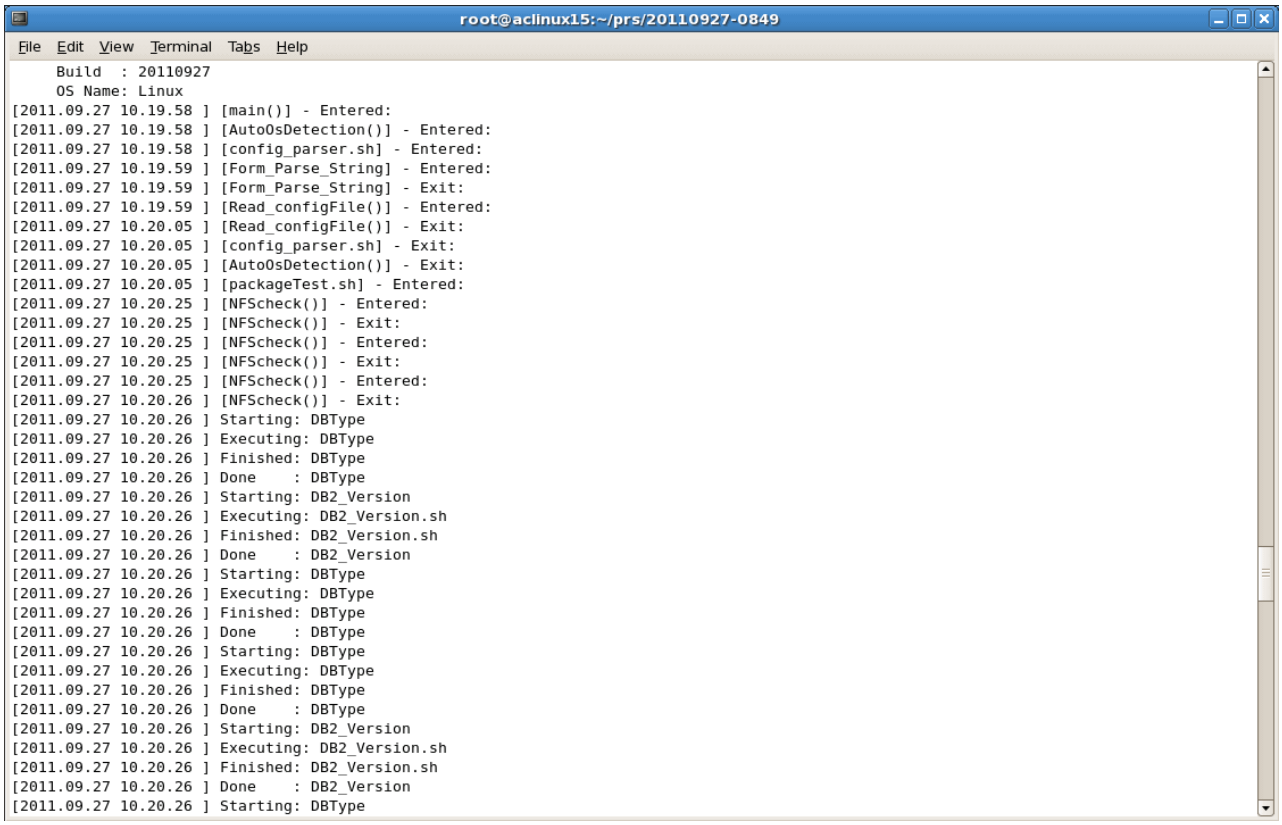


Abbildung 15. Datei "prs.trc" auf UNIX-Systemen

Probleme bei der Ausführung

Sie können die Checkliste für Probleme bei der Ausführung verwenden, um Fehler zu beheben, die bei der Ausführung von Prerequisite Scanner auftreten können.

Führen Sie das Script von Prerequisite Scanner mit den optionalen Eingabeparametern **debug** und **trace** aus, um das Debugging von Problemen zu unterstützen.

Tabelle 13. Checkliste für Probleme bei der Ausführung

Prüfung	Problem
☐	Wenn Sie den optionalen Parameter **outputDir** in der Befehlszeile angeben und das Ausgabeverzeichnis nicht vorhanden ist, erstellt Prerequisite Scanner das Verzeichnis. Sie müssen Schreibberechtigungen besitzen, um das Ausgabeverzeichnis, in dem Prerequisite Scanner die Dateien speichert, zu erstellen oder zu beschreiben. Wenn Sie keine Schreibberechtigungen besitzen, wird die folgende Fehlernachricht in der Befehlszeilenschnittstelle ausgegeben: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`
☐	Stellen Sie vor der Ausführung von Prerequisite Scanner sicher, dass die Speicherkapazität der Platte, auf der Sie Prerequisite Scanner ausführen und die Ergebnisse im Ausgabeverzeichnis speichern möchten, nicht erschöpft ist. Andernfalls wird die folgende Fehlernachricht in der Befehlszeilenschnittstelle ausgegeben: `ERROR: Cannot create files in output directory ips_output_dir. Exit.`

Tabelle 13. Checkliste für Probleme bei der Ausführung (Forts.)

Prüfung	Problem
☐	Wenn Prerequisite Scanner den Rückgabecode 2 generiert, ist möglicherweise ein Fehler in der Scriptsyntax oder im Collector aufgetreten. Überprüfen Sie die Ursachen, die diesem Fehlercode zugeordnet sind. Wenn ein Scriptsyntaxfehler aufgetreten ist, führen Sie Prerequisite Scanner mit der richtigen Syntax erneut aus.

Zugehörige Konzepte:

Das Schreiben von Nachrichten in Protokolldateien ist auf UNIX-Systemen standardmäßig inaktiviert. Sie können die Debug- und Tracefunktionen mit den Eingabeparametern **debug** und **trace** aktivieren. Der Scanner schreibt die Debug-und Tracedaten in verschiedene Protokolldateien und verwendet Zeitmarken, um die Start- und Endzeiten der Schritte oder Funktionen zu kennzeichnen. Sie können beide Dateien verwenden, um ein bestimmtes Problem, eine bestimmte Funktion oder eine bestimmte Prüfung der Voraussetzungen zu korrelieren und die Fehlerbehebung durchzuführen.

Prerequisite Scanner generiert Rückgabecodes, die von den Ergebnissen des Scans und davon abhängig sind, ob der Scanner aufgrund von Fehlern beendet werden muss. Diese Rückgabecodes werden in die Protokolldateien geschrieben.

Das Script `prereq_checker` führt IBM Prerequisite Scanner aus und überprüft die Voraussetzungen basierend auf den Parametern, die Sie angeben, wenn Sie das Script ausführen.

Rückgabecodes

Prerequisite Scanner generiert Rückgabecodes, die von den Ergebnissen des Scans und davon abhängig sind, ob der Scanner aufgrund von Fehlern beendet werden muss. Diese Rückgabecodes werden in die Protokolldateien geschrieben.

Prerequisite Scanner generiert Rückgabecodes basierend auf einer Gruppe definierter Ergebnisse:

Rückgabecode	Beschreibung
`0`	Gibt `0` zurück, wenn Prerequisite Scanner erfolgreich ausgeführt wird und alle Scanergebnisse `PASS` lauten.
`1`	Gibt `1` zurück, wenn Prerequisite Scanner erfolgreich ausgeführt wird, aber eine oder mehrere Prüfungen der Voraussetzungen `FAIL` zurückgeben.
`2`	Gibt `2` zurück, wenn Prerequisite Scanner nicht erfolgreich ausgeführt wird und aufgrund eines Fehlers, der wie folgt kategorisiert ist, beendet werden muss: • Fehler in der Scriptsyntax • Collectorfehler • Sonstige Fehler

Fehler in der Scriptsyntax

Prerequisite Scanner kann beendet werden, weil einer der folgenden Syntaxfehler bei der Ausführung des Scripts auftritt:

- Der Eingabeparameter **Product_Code** ist nicht gültig. Er wurde beispielsweise nicht gefunden, oder er hat kein unterstütztes Format.
- Das Muster für die Eingabeparameter **Product_Code** und **Product_Version** ist ungültig. Es wurde beispielsweise mehr als nur Code und Version in Anführungszeichen gesetzt, oder das Muster ist nicht in Anführungszeichen eingeschlossen.
- Der Eingabeparameter **Product_Version** ist ungültig. Die Produktversion besteht beispielsweise nicht ausschließlich aus numerischen Zeichen.
- Es wurden keine Eingabeparameter in der Befehlszeilenschnittstelle eingegeben.
- Die in der Befehlszeilenschnittstelle eingegebene Syntax ist ungültig. Es wurde beispielsweise ein nicht unterstütztes Befehlszeilenargument eingegeben.
- Der erforderliche Eingabeparameter **Product_Code** wurde nicht eingegeben.

Collectorfehler

Prerequisite Scanner kann beendet werden, weil einer der folgenden Collectorfehler auftritt:

- Die temporäre Ergebnisdatei des Collectors wurde nicht im Verzeichnis *ips_output_dir*/temp gefunden.
- Die Scriptdatei des Collectors wurde nicht ordnungsgemäß ausgeführt.

Sonstige Fehler

Prerequisite Scanner kann beendet werden, weil der Benutzer keine Schreibberechtigung für das Ausgabeverzeichnis *ips_output_dir* hat.

Zugehörige Konzepte:

IBM Prerequisite Scanner erzeugt Ausgaben für die folgenden Bildschirm- und lesbaren Dateiformate: Ausgabe in der Befehlszeilenschnittstelle, Debug- und Traceprotokolldateien, Text- und XML-Dateien für die Ergebnisse.

Anhang A. Produktcodereferenzen

IBM Prerequisite Scanner verwendet einen Mehrzeichencode, *product_code*, um das Produkt, die jeweilige unterstützte Plattform und die Version des Betriebssystems zu identifizieren. Die Datei *ips_root*/codename.cfg enthält die Name/Wert-Paare, die den Produktcode für das Produkt, dessen unterstützte Plattform und dessen Version des Betriebssystems darstellen.

In Tabelle 14 ist der aktuelle Satz vordefinierter Produktcodes beschrieben.

Einschränkung: IBM Tivoli Monitoring und Tivoli Composite Application Manager haben vordefinierte Produktcodes, die Prerequisite Scanner als reservierte Codes betrachtet. Diese Codes dürfen nicht als Produktcodes von Prerequisite Scanner verwendet werden, es sei denn, sie verweisen auf die zugehörigen Agenten von IBM Tivoli Monitoring und Tivoli Composite Application Manager. Weitere Informationen zum Hinzufügen der Produktcodes finden Sie im technischen Hinweis zu den Produktcodes für ITM 6.X.

Einschränkung: Nur UNIX: Wenn Sie den Wert für den Produktcode in der Datei eingeben, vermeiden Sie die Verwendung von `for`. Dies ist ein reserviertes Wort und kann sich auf die Ausführung von Prerequisite Scanner auswirken.

Tabelle 14. Vordefinierte Produktcodes

Vordefinierter Produktcode	Plattform	Produktversion, Plattform, Betriebssystem
ADE	Alle	Autonomic Deployment Engine
BSM	Alle	Tivoli Business Service Manager
CDB	Alle	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 Bit}
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 Bit)
COH	Linux	Tivoli Provisioning Manager for Red Hat Enterprise Linux 5 System z (64 Bit)
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

Tabelle 14. Vordefinierte Produktcodes (Forts.)

Vordefinierter Produktcode	Plattform	Produktversion, Plattform, Betriebssystem
CON	Linux	Tivoli Provisioning Manager Version 7.2 for SUSE zSeries 11
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	Alle	Prerequisite Scanner (Demo)
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	Alle	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	Alle	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	Alle	Tivoli Composite Application Manager Agent for WebSphere 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	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Reporter
KTU	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Transaction Collector
KT3	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console
KT4	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time

Tabelle 14. Vordefinierte Produktcodes (Forts.)

Vordefinierter Produktcode	**Plattform**	**Produktversion, Plattform, Betriebssystem**
KT5	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time
KT6	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Robotic Response Time
KZE	Alle	Tivoli zEnterprise Monitoring Agent
LCM	Windows	Tivoli License Compliance Manager
	UNIX	Tivoli License Compliance Manager for UNIX
NCI	Alle	Tivoli Netcool/Impact
NOC	Alle	Tivoli Netcool/OMNIbus - Serverkomponenten und Desktopkomponenten
NOD	Alle	Tivoli Netcool/OMNIbus - Desktopkomponenten
NOS	Alle	Tivoli Netcool/OMNIbus - Serverkomponenten
PAE	Alle	Tivoli Process Automation Engine
TAD	Windows	Tivoli Asset Discovery for Distributed
	UNIX	Tivoli Asset Discovery for Distributed for UNIX
TCR	Alle	Tivoli Common Reporting
TPM	Alle	Tivoli Provisioning Manager

Anhang B. Referenzinformationen zu Konfigurationsdateien

IBM Prerequisite Scanner stellt eine vordefinierte Gruppe von Konfigurationsdateien bereit, die Sie bearbeiten können. Diese Dateien befinden sich im Verzeichnis *ips_root*/UNIX_Linux bzw. *ips_root*/Windows. Die Dateien haben die Erweiterung .cfg.

Tabelle 15 listet die momentan unterstützten vordefinierten Konfigurationsdateien auf.

Tabelle 15. Vordefinierte Konfigurationsdateien

Konfigurationsdatei	Plattform	Produktversion, Plattform, Betriebssystem
ADE_01040000.cfg	Alle	Autonomic Deployment Engine Version 1.4
BSM_04210000.cfg	Alle	Tivoli Business Service Manager Version 4.2.1
BSM_06100000.cfg	Alle	Tivoli Business Service Manager Version 6.1
CDB_06220000.cfg	Alle	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 Version 5.3.0.0 {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 Bit)
COH_07200000.cfg	Linux	Tivoli Provisioning Manager Version 7.2 for Red Hat Enterprise Linux 5 System z (64 Bit)
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	Alle	Prerequisite Scanner (Demo)
DMO_01000000.cfg	Alle	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	Alle	Tivoli Enterprise Portal Client Version 6.2.2
KCQ_06200000.cfg	Windows	Tivoli Enterprise Portal Server Version 6.2

Tabelle 15. Vordefinierte Konfigurationsdateien (Forts.)

Konfigurationsdatei	Plattform	Produktversion, Plattform, Betriebssystem
KCQ_06210000.cfg	UNIX	Tivoli Enterprise Portal Server Version 6.2.2
KCQ_06220000.cfg	Alle	Tivoli Enterprise Portal Server Version 6.2.2
KHD_06200000.cfg	Windows	Warehouse Proxy Agent Version 6.2
KHD_06210000.cfg	Alle	Warehouse Proxy Agent Version 6.2.1
KHD_06220000.cfg	Alle	Warehouse Proxy Agent Version 6.2.2
KHE_06220000.cfg	UNIX	Warehouse Proxy Agent Version 6.2.2
KIS_07200000.cfg	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Internet Service Monitoring Version 7.2
KIS_07300000.cfg	Alle	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	Alle	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	Alle	Tivoli Enterprise Monitoring Server Version 6.2.1
KMS_06220000.cfg	Alle	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	Alle	Tivoli Composite Application Manager Agent for WebSphere Message Broker Version 7.1
KSY_06200000.cfg	Windows	Summarization and Pruning Agent Version 6.2
KSY_06210000.cfg	Alle	Summarization and Pruning Agent Version 6.2.1
KSY_06220000.cfg	Alle	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	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Application Management Console Version 7.3
KT4_07300000.cfg	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Client Response Time Version 7.3

Tabelle 15. Vordefinierte Konfigurationsdateien (Forts.)

Konfigurationsdatei	Plattform	Produktversion, Plattform, Betriebssystem
KT5_07300000.cfg	Alle	Tivoli Composite Application Manager (ITCAM) for Transactions: Web Response Time Version 7.3
KT6_07300000.cfg	Alle	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	Alle	Tivoli Monitoring Agent for DB2 Version 6.2
KUD_06210000.cfg	Alle	Tivoli Monitoring Agent for DB2 Version 6.2.1
KUD_06220000.cfg	Alle	Tivoli Monitoring Agent for DB2 Version 6.2.2
KZE_06020300.cfg	Alle	Tivoli zEnterprise Monitoring Agent Version 6.2.3
LCM_01000000.cfg	Alle	Tivoli License Compliance Manager Version 1.0
LCM_02300000.cfg	Alle	Tivoli License Compliance Manager Version 2.3
NCI_06100000.cfg	Alle	Tivoli Netcool/Impact Version 6.1
NOC_07310000.cfg	Alle	Tivoli Netcool/OMNIbusVersion 7.3.1 - Serverkomponents und Desktopkomponenten
NOD_07310000.cfg	Alle	Tivoli Netcool/OMNIbus Version 7.3.1 - Desktopkomponente
NOS_07310000.cfg	Alle	Tivoli Netcool/OMNIbus Version 7.3.1 - Serverkomponenten
PAE_07500000.cfg	Alle	Tivoli Process Automation Engine
TAD_07200000.cfg	Alle	Tivoli Asset Discovery for Distributed Version 7.2
TAD_07220000.cfg	Alle	Tivoli Asset Discovery for Distributed Version 7.2.2
TCR_02010100.cfg	Alle	Tivoli Common Reporting
TPM_07210000.cfg	Alle	Tivoli Provisioning Manager Version 7.2.1

Anhang C. Referenzinformationen zu vorausgesetzten Eigenschaften

In diesen Referenzinformationen sind die vorausgesetzten Basiseigenschaften für jede vordefinierte Kategorie von Hardware- und Softwarevoraussetzungen beschrieben.

In Tabelle 16 sind die vordefinierten Kategorien von Hardware- und Softwarevoraussetzungen beschrieben.

Tabelle 16. Vordefinierte Kategorien für vorausgesetzte Eigenschaften

Datenkategorie	**Beschreibung**	**Erforderliche Präfix-ID**	**Referenzinformationen**
Allgemein	Die allgemeinen Dateneigenschaften überprüfen allgemeine Voraussetzungen wie die Prozessorgeschwindigkeit, den Arbeitsspeicher, den Plattenspeicherplatz und den temporären Speicherplatz.	Keine	„Allgemeine Dateneigenschaften" auf Seite 92
Autonomic Deployment Engine	Die Dateneigenschaften für Autonomic Deployment Engine überprüfen Voraussetzungen von Autonomic Deployment Engine, wie z. B. die Installationseinheit.	`de`	„Dateneigenschaften für Autonomic Deployment Engine" auf Seite 97
Installierte Software	Die Dateneigenschaften für installierte Software überprüfen die Voraussetzungen für installierte Software, wie z. B. die in der Windows-Registry registrierten Programme. Außerdem wird geprüft, ob cygwin und gskit installiert sind.	Keine	„Dateneigenschaften für installierte Software" auf Seite 113
Benutzer	Die Dateneigenschaften für Benutzer überprüfen Voraussetzungen für Benutzer, z. B., ob der angemeldete Benutzer Administratorberechtigungen hat oder ob er der Rootbenutzer ist.	`user`	„Dateneigenschaften für Benutzer" auf Seite 114
Betriebssystem	Die Dateneigenschaften für Betriebssysteme überprüfen die Voraussetzungen für das Betriebssystem, z. B. Version, Architektur, Gesamtspeicher, verfügbaren Hauptspeicher und physischen Gesamthauptspeicher.	`os`	„Dateneigenschaften für Betriebssysteme" auf Seite 101
Konnektivität	Die Dateneigenschaften für die Konnektivität überprüfen die Voraussetzungen für die Konnektivität, z. B., ob Telnet aktiv ist und zu welchen IP-Adressen und Ports der Scanner eine Verbindung herstellen kann.	Keine	„Dateneigenschaften für die Konnektivität" auf Seite 98
Netz	Die Dateneigenschaften für Netze überprüfen die allgemeinen Netzvoraussetzungen für alle Plattformen, z. B., ob Ports verfügbar sind.	`network`	„Dateneigenschaften für Netze" auf Seite 99
Windows-Netz	Die Dateneigenschaften für Windows-Netze überprüfen Netzvoraussetzungen, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften.	`network`	„Dateneigenschaften für Windows-Netze" auf Seite 114

Tabelle 16. Vordefinierte Kategorien für vorausgesetzte Eigenschaften (Forts.)

Datenkategorie	Beschreibung	Erforderliche Präfix-ID	Referenzinformationen
UNIX-Netz	Die Dateneigenschaften für UNIX-Netze überprüfen Netzvoraussetzungen, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften.	`network`	„Dateneigenschaften für UNIX-Netze" auf Seite 114
Internet Explorer	Die Dateneigenschaften für Internet Explorer überprüfen die Voraussetzungen für Microsoft Internet Explorer, wie z. B. die Version.	`internetExplorer`	„Dateneigenschaften für Internet Explorer" auf Seite 99
Datenbankserver, DB2	Die Dateneigenschaften für DB2 überprüfen die Voraussetzungen für DB2, z. B. die Version.	`DB2`	„Dateneigenschaften für DB2" auf Seite 98
Datenbankserver, MS SQL	Die Dateneigenschaften für MS SQL Server überprüfen die Voraussetzungen für MS SQL Server, wie z. B. die Version.	`mssql`	„Dateneigenschaften für MS SQL Server" auf Seite 98
Datenbankserver, Oracle	Die Dateneigenschaften für Oracle überprüfen die Oracle-Voraussetzungen, wie z. B. die Version.	`Oracle`	„Dateneigenschaften für Oracle" auf Seite 100
Umgebungsvariablen	Dies Umgebungsvariablen überprüfen Voraussetzungen für Umgebungsvariablen, z. B., ob die Umgebungsvariable gesetzt ist.	`env`	„Dateneigenschaften für Umgebungsvariablen" auf Seite 115

Allgemeine Dateneigenschaften

Die allgemeinen Dateneigenschaften überprüfen allgemeine Voraussetzungen wie die CPU-Geschwindigkeit, den Arbeitsspeicher, den Plattenspeicherplatz und den temporären Speicherplatz. Für Windows-Systeme wird das Hauptscript von IBM Prerequisite Scanner verwendet. Für UNIX-Systeme werden das Hauptscript von Prerequisite Scanner und der allgemeine Collector *IPS-Stammverzeichnis*`/Unix_Linux/common.sh` verwendet.

In Tabelle 17 sind die allgemeinen vorausgesetzten Dateneigenschaften beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert keine Präfix-ID.

Tabelle 17. Allgemeine vorausgesetzte Dateneigenschaften

Vorausgesetzte Eigenschaft	Plattformen	Beschreibung	Gültige Werte
`CPU Name`	Alle	Der Name der CPU; wird nur für Anzeigezwecke in den Ergebnissen verwendet.	Nicht zutreffend
`CpuArchitecture`	UNIX	Die Architektur des Betriebssystems.	Zeichenfolge mit mehreren unterstützten Werten, die durch Kommas getrennt sind, z. B.: `x86_64,s390x,ppc64,AMD64`

Tabelle 17. Allgemeine vorausgesetzte Dateneigenschaften (Forts.)

Vorausgesetzte Eigenschaft	Plattformen	Beschreibung	Gültige Werte

DBType	Alle	Überprüft den Typ des auf der Maschine installierten Datenbankservers.
		
		Nur für Oracle auf UNIX-Systemen: Der Collector erwartet, dass die Umgebungsvariablen ORACLE_BASE und ORACLE_HOME in der Datei $HOME/.profile gesetzt sind, z. B.:
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		$HOME muss auf /home/oracle, das Ausgangsverzeichnis für den Oracle-Benutzer, gesetzt werden.
			Gültige Typen für den Wert:
			• Zeichenfolge, die einen beliebigen Typ des Datenbankservers darstellt, z. B.:
			`any`
			• Zeichenfolge, die den Typ des Datenbankservers darstellt, z. B.:
			`Oracle`
			• regex{*str*}, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für den Datenbankservertyp darstellt, z. B.:
			`regex{.*MSSQL.*|DB2.*}`
			
			Überprüft, ob der Datenbankservertyp auf Windows-Systemen MS SQL oder DB2 ist.
			• Zeichenfolge, die keinen Typ des Datenbankservers darstellt, z. B.:
			`unknown`

DBTypeDetails	Alle	Die Typen des auf der Maschine installierten Datenbankservers.
		
		Nur für Oracle auf UNIX-Systemen: Der Collector erwartet, dass die Umgebungsvariablen ORACLE_BASE und ORACLE_HOME in der Datei $HOME/.profile gesetzt sind, z. B.:
		`export ORACLE_BASE=/home/oracle/app/oracle/product/11.2.0/`
		`export ORACLE_HOME=/home/oracle/app/oracle/product/11.2.0/dbhome_1`
		
		$HOME muss auf /home/oracle, das Ausgangsverzeichnis für den Oracle-Benutzer, gesetzt werden.
		
		Die vorausgesetzte Eigenschaft schreibt die Details zum Datenbankservertyp, d. h. den Datenbankservertyp, die Installationsposition und die Version, in die Datei result.txt. Die Details mehrerer Datenbankservertypen werden durch Semikolons getrennt.
			Gültige Typen für den Wert:
			• Zeichenfolge, die einen beliebigen Typ des Datenbankservers darstellt, z. B.:
			`any`
			• Zeichenfolge, die einen einzigen Typ des Datenbankservers darstellt, z. B.:
			`DB2`
			• regex{*str*}, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für den Datenbankservertyp darstellt, z. B.:
			`regex{.*MSSQL.*|DB2.*}`
			
			Überprüft, ob der Datenbankservertyp auf Windows-Systemen MS SQL oder DB2 ist.

Tabelle 17. Allgemeine vorausgesetzte Dateneigenschaften (Forts.)

Vorausgesetzte Eigenschaft	Plattformen	Beschreibung	Gültige Werte

`Disk`	Windows	Der freie Plattenspeicherplatz mit den folgenden optionalen Qualifikationsattributen.
		• Attribut `dir` zum Bestimmen des zu prüfenden Verzeichnispfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
			Gültige Typen für den Wert:
			• Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[dir:`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			Beispiel:
			`Disk= [dir:C:\Program Files\IBM\SQLLIB, unit:MB]1431`
			• Numerisches Format in MB oder GB:
			`disk_space``MB|GB`
			Beispiel:
			`Disk=250MB`

`Disk`	UNIX	Der freie Plattenspeicherplatz.	Numerisches Format in GB oder MB, z. B.: `2GB`
`intel.cpu`	Alle	Die CPU-Geschwindigkeit für den Intel-Prozessor.	Numerisches Format in GHz und nur unter Windows auch in MHz, z. B.: `2GHz`
`Memory`	Alle	Die physische Gesamthauptspeicherkapazität, die momentan auf der Maschine verfügbar ist. **Tipp:** Überprüft anhand der vordefinierten vorausgesetzten Eigenschaften in der Betriebssystemkategorie die physische und die virtuelle Hauptspeicherkapazität gesondert.	Numerisches Format in GB oder MB, z. B.: `300MB`

Tabelle 17. Allgemeine vorausgesetzte Dateneigenschaften (Forts.)

Vorausgesetzte Eigenschaft	Plattformen	Beschreibung	Gültige Werte

`OS Version`	Alle	Der vollständige Name und die Version des Betriebssystems, das auf der Maschine ausgeführt wird. Alternativ können Sie einen regulären Ausdruck verwenden, um eine Zeichenfolge zu übergeben, die mehrere Varianten eines Betriebssystems darstellt.
		Tipp: Verwenden Sie diese vorausgesetzte Eigenschaft zusammen mit `os.servicePack` und `os.architecture`, um das aktuelle Service-Pack und die Systemarchitektur zu überprüfen.
			Gültige Typen für den Wert:
			• Zeichenfolge, die mehrere Versionen darstellen kann, wobei die einzelnen Versionen durch Kommas voneinander getrennt werden, z. B.:
			`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`
			Einschränkung: Auf Windows-Systemen wird der Platzhalter `*` nur in einem regulären Ausdruck unterstützt
			• `regex{`*`str`*`}`, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für die Version darstellt, z. B.:
			`regex{Windows 200[3-8]}`
			Prüft, ob das tatsächliche Betriebssystem einer Version von Windows 2003 bis Windows 2008 entspricht.
			`regex{Red Hat*.*}`
			Prüft, ob das tatsächliche Betriebssystem eine Variante von Red Hat Linux ist.
			Anmerkung: In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

`numCPU`	Alle	Die Anzahl der Kerne oder unabhängigen Prozessoren im Computer. Wenn das Tool den Computer scannt und keine Kerne findet oder Prozessoren findet, von denen keiner ein Kern ist, gibt er ein Ergebnis "Not found" zurück.	Zahl, z. B. `4`
`risc.cpu`	UNIX	Die CPU-Geschwindigkeit für einen RISC-Prozessor.	Numerisches Format in GHz, z. B.: `1.4GHz`
`Temp`	UNIX	Der verfügbarer Plattenspeicherplatz für das angegebene Dateisystem *Temp*.	Numerisches Format in GB oder MB, z. B.: `300MB`

Zugehörige Konzepte:

Die Überprüfung der vorausgesetzten Eigenschaft `Memory` in Prerequisite Scanner variiert je nachdem, ob ein Agent von Tivoli Monitoring oder Tivoli Composite Application Manager bereits auf dem Computer ausgeführt wird oder nicht.

Zugehörige Verweise:

Die Dateneigenschaften für Betriebssysteme überprüfen die Voraussetzungen für das Betriebssystem, z. B. Version, Architektur, Gesamtspeicher, verfügbaren Hauptspeicher und physischen Gesamthauptspeicher. Für Windows-Systeme werden die VBScript-Collector für Betriebssysteme im Verzeichnis *ips_root*`/lib` mit der Präfix-ID `os` im Dateinamen verwendet. Für UNIX-Systeme werden die UNIX-Betriebssystemcollector im Verzeichnis *ips_root*`/UNIX_Linux` mit der Präfix-ID `os` im Dateinamen verwendet.

Systemverhalten für die vorausgesetzte Eigenschaft "Memory" und Agenten von Tivoli Monitoring

Die Überprüfung der vorausgesetzten Eigenschaft `Memory` in Prerequisite Scanner variiert je nachdem, ob ein Agent von Tivoli Monitoring oder Tivoli Composite Application Manager bereits auf dem Computer ausgeführt wird oder nicht.

Wenn ein Agent installiert ist, verwendet Prerequisite Scanner einen erwarteten Wert für die vorausgesetzte Eigenschaft `Memory`, der auf der Differenz zwischen den erwarteten Werten aus der neuen und der vorhandenen Konfigurationsdatei basiert, wenn die vorhandene Konfigurationsdatei noch auf dem Computer vorhanden ist. Andernfalls wird der erwartete Wert wie üblich behandelt.

Wenn Sie Prerequisite Scanner ausführen, um die Voraussetzungen für einen Agenten von Tivoli Monitoring zu prüfen, der aktualisiert oder erneut installiert wird, wird zunächst geprüft, ob der Agent bereits auf dem Computer ausgeführt wird. Wird der Agent auf dem Computer ausgeführt, sucht Prerequisite Scanner die Konfigurationsdatei, die der vorhandenen Version des aktiven Agenten zugeordnet ist. Das folgende Verhalten richtet sich nach dem Ergebnis dieser Suche:

- Wird die Konfigurationsdatei nicht gefunden, geht Prerequisite Scanner davon aus, dass die Zielumgebung noch nicht gescannt wurde. Deshalb verwendet Prerequisite Scanner den erwarteten Wert für die vorausgesetzte Eigenschaft `Memory`, die in der neuen Konfigurationsdatei angegeben ist, was dem Standardverhalten entspricht. Prerequisite Scanner schreibt diesen erwarteten Wert in die Ergebnisausgabe.
- Wird die Konfigurationsdatei gefunden, vergleicht Prerequisite Scanner den erwarteten Wert der vorausgesetzten Eigenschaft `Memory` der vorhandenen Version mit dem erwarteten Wert in der Konfigurationsdatei der neuen Version. Gibt es eine Differenz zwischen den Werten und ist der neue Wert höher als der vorhandene erwartete Wert, setzt Prerequisite Scanner diese Differenz als erwarteten Wert. Prerequisite Scanner schreibt diese Differenz als erwarteten Wert in die Ergebnisausgabe. Angenommen, die Konfigurationsdatei für den Agenten der Version 1 gibt 1 GB als erwarteten Wert an. Die neue Konfigurationsdatei für den Agenten der Version 2 gibt 1.5 GBs als erwarteten Wert an. Deshalb verwendet und schreibt Prerequisite Scanner die Differenz von 0.5 GB als erwarteten Wert.

Dateneigenschaften für Autonomic Deployment Engine

Die Dateneigenschaften für Autonomic Deployment Engine überprüfen Voraussetzungen von Autonomic Deployment Engine, wie z. B. die Installationseinheit. Für Windows-Systeme werden die Collector von Autonomic Deployment Engine im Verzeichnis *IPS-Stammverzeichnis*/lib/ mit dem Präfix de im Dateinamen verwendet. Für UNIX-Systeme werden die UNIX-Collector von Autonomic Deployment Engine im Verzeichnis *IPS-Stammverzeichnis*/UNIX_Linux mit dem Präfix de im Dateinamen verwendet.

In Tabelle 18 sind die vorausgesetzten Eigenschaften beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID de.

Tabelle 18. Dateneigenschaften für Autonomic Deployment Engine

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
de.installed	Alle	Prüft, ob die Version installiert ist.	Boolescher Wert, z. B.: true|false
de.installation-Unit	Alle	Überprüft, ob die angegebene Installationseinheit mit dem Befehl **listIU** -v installiert wurde.	Gültige Typen für den Wert: • Zeichenfolge zur Darstellung einer einzelnen Installationseinheit, z. B. der Installationseinheit für Tivoli Integrated Portal: C37109911C8A11D98E1700061BDE7AEA, B24209911C8A11D98E1700061BDE7AEA • Zeichenfolge für die Darstellung mehrerer Installationseinheiten, z. B.: 5FFE79F918DF3BA0D67511FD3F7C358E • regex {str}, ein regulärer Ausdruck mit dem Eingabeparameter str, der das Suchmuster für die Installationseinheit, die Version und den Installationspfad darstellt. Um beispielsweise die Installationseinheit und die Version von WebSphere Application Server und den Installationspfad für Tivoli Integrated Portal zu suchen, ist das Suchmuster wie folgt: • regex{.*C00DA95AFD9B7E0397153CD944B5A255.*6.1.0.2100.*SIU eWAS.*C:\\IBM\\tivoli\\tip.*} **Anmerkung:** Sie können auch eine Umgebungsvariable für den Installationspfad verwenden, z. B., indem Sie den Pfad durch die Umgebungsvariable TIPHOME ersetzen. In diesem Fall ist das Suchmuster folgendes: regex{.*C00DA95AFD9B7E0397153CD944B5A255.*6.1.0.2100.*SIU eWAS.*%TIPHOME%.*} • Meherere regex {str}-Argumente für die Darstellung mehrerer Prüfungen, z. B.: regex{.*C37109911C8A11D98E1700061BDE7AEA.*}, regex{.*B24209911C8A11D98E1700061BDE7AEA.*}

Dateneigenschaften für die Konnektivität

Die Dateneigenschaften für die Konnektivität überprüfen die Voraussetzungen für die Konnektivität, z. B., ob Telnet aktiv ist und zu welchen IP-Adressen und Ports der Scanner eine Verbindung herstellen kann. Für Windows-Systeme wird der Konnektivitätscollector *IPS-Stammverzeichnis*/lib/connectivity_plug.vbs verwendet. Für UNIX-Systeme werden das Hauptscript von IBM Prerequisite Scanner und der Konnektivitätscollector *IPS-Stammverzeichnis*/Unix_Linux/connectivity_plug.sh verwendet. Die Ausgabe wird nur in der Debugprotokolldatei ausgegeben.

Dateneigenschaften für DB2

Die Dateneigenschaften für DB2 überprüfen die Voraussetzungen für DB2, z. B. die Version. Für Windows-Systeme wird der DB2-Collector *IPS-Stammverzeichnis*/lib/db2_version_plug.bat verwendet. Für UNIX-Systeme werden die UNIX-DB2-Collector im Verzeichnis *IPS-Stammverzeichnis*/UNIX_Linux mit dem Präfix db2 im Dateinamen verwendet.

In Tabelle 19 sind die vorausgesetzten Eigenschaften für DB2 beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID DB2.

Tabelle 19. Dateneigenschaften für DB2

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
DB2 Version	Alle	Die Version von DB2, die derzeit auf dem System installiert ist.	Zeichenfolge, z. B.: v9.5.100.179FP4
db2.home.space	UNIX	Der verfügbare Plattenspeicherplatz für das DB2-Ausgangsverzeichnis.	Numerisches Format in GB, z. B.: 8GB

Dateneigenschaften für MS SQL Server

Die Dateneigenschaften für MS SQL Server überprüfen die Voraussetzungen für MS SQL Server, wie z. B. die Version und die Position. Für Windows-Systeme werden die MS-SQL-Server-Collector im Verzeichnis *IPS-Stammverzeichnis*/Windows mit dem Präfix mssql im Dateinamen verwendet.

In Tabelle 20 sind die vorausgesetzten Eigenschaften für MS SQL Server beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID mssql.

Tabelle 20. Dateneigenschaften für MS SQL Server

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
mssql.Client	Windows	Überprüft die Version des MS-SQL-Clients, der momentan auf dem System installiert ist.	Der erwartete Zeichenfolgewert kann mehrere durch Kommas getrennte Versionen enthalten, z. B.: 10.50.1600.1 **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

Tabelle 20. Dateneigenschaften für MS SQL Server (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`mssql.Server`	Windows	Überprüft die Version des MS-SQL-Servers, der momentan auf dem System installiert ist.	Der erwartete Zeichenfolgewert kann mehrere durch Kommas getrennte Versionen enthalten, z. B.: `10.50.1600.1` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`mssql.Server.Location`	Windows	Überprüft das Ausgangsverzeichnis des MS-SQL-Datenbankservers.	Zeichenfolge, z. B.: `any`

Dateneigenschaften für Internet Explorer

Die Dateneigenschaften für Internet Explorer überprüfen die Voraussetzungen für Microsoft Internet Explorer, wie z. B. die Version. Es wird der Internet-Explorer-Collector *IPS-Stammverzeichnis*`/lib/internetExplorer_plug.vbs` verwendet.

In Tabelle 21 sind die vorausgesetzten Eigenschaften für Internet Explorer beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `internetExplorer`.

Tabelle 21. Dateneigenschaften für Internet Explorer

Vorausgesetzte Eigenschaft	Beschreibung	Gültige Werte
`internetExplorer.version`	Die Version von Internet Explorer, die auf der Maschine installiert ist.	Numerisches Format, z. B. `7.0+` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

Dateneigenschaften für Netze

Die Dateneigenschaften für Netze überprüfen die allgemeinen Netzvoraussetzungen für alle Plattformen, z. B., ob Ports verfügbar sind. Es werden die Netzcollector im Verzeichnis *IPS-Stammverzeichnis*`/lib` mit der Präfix-ID `network` im Dateinamen verwendet.

In Tabelle 22 auf Seite 100 sind die vorausgesetzten Eigenschaften für Netze beschrieben, die für alle Plattformen gelten. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `network`.

Tabelle 22. Dateneigenschaften für Netze

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte

`network.availablePorts.`
app_type
	Alle	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Port bzw. Portbereich für den Anwendungstyp *app_type* gültig ist. Sie können prüfen, welche Ports nicht überwacht werden, z. B.:
		• `network.availablePorts.`
		`DB2` überprüft die Ports für den DB2-Datenbankserver, wobei *app_type* durch `DB2` ersetzt wird.
		• `network.availablePorts.`
		`WAS` überprüft die Ports für WebSphere Application Server, wobei *app_type* durch `WAS` ersetzt wird.
			Positive ganze Zahlen, z. B.:
			• `network.availablePorts.`
			`DB2 = 50000-50005`
			• `network.availablePorts.`
			`WAS = 8080`
			Anmerkung: In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

`network.portsInUse.`
app_type
	Alle	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Port bzw. Portbereich für den Anwendungstyp *app_type* im Gebrauch ist. Sie können prüfen, welche Ports überwacht werden, z. B.:
		• `network.availablePorts.`
		`DB2` überprüft die Ports für den DB2-Datenbankserver, wobei *app_type* durch `DB2` ersetzt wird.
		• `network.availablePorts.`
		`WAS` überprüft die Ports für WebSphere Application Server, wobei *app_type* durch `WAS` ersetzt wird.
			Positive ganze Zahlen, z. B.:
			• `network.portsInUse.`
			`DB2 = 50900-50905`
			• `network.portsInUse.`
			`WAS = 8080`
			Anmerkung: In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

`network.validate` `HostsFile`	Windows	Prüft, ob alle Hostmaschinen , die in der Datei "hosts" aufgelistet sind, das folgende Format haben: *IP_Address Host_Name Short_Name* Erläuterungen: • *IP_Address* ist die IP-Adresse für den Computer, z. B. `127.0.0.1`. • *Host_Name* ist der vollständig qualifizierte Hostname des Computers, z. B. `localhost.localdomain`. • *Short_Name* ist der Kurzname des Computers, z. B. `localhost`.	Boolescher Wert, z. B. `True`

Dateneigenschaften für Oracle

Die Dateneigenschaften für Oracle überprüfen die Oracle-Voraussetzungen, wie z. B. die Version. Für Windows-Systeme wird der Oracle-Collector verwendet. Für UNIX-Systeme werden die UNIX-Oracle-Collector im Verzeichnis *IPS-Stammverzeichnis*`/UNIX_Linux` mit dem Präfix `oracle` im Dateinamen verwendet. Für Window-Systeme werden die Windows-Oracle-Collector im Verzeichnis *IPS-Stammverzeichnis*`/lib` mit dem Präfix `oracle` im Dateinamen verwendet.

In Tabelle 23 sind die vorausgesetzten Eigenschaften für Oracle beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `oracle`.

Tabelle 23. Dateneigenschaften für Oracle

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`ORACLE Version`	Windows	Überprüft die Version von Oracle, die momentan auf dem System installiert ist.	Der erwartete Zeichenfolgewert kann mehrere durch Kommas getrennte Versionen enthalten, z. B.: `9.2, 10.1, 10.2`
`oracle.Client`	Alle	Überprüft die Version des Oracle-Clients, der momentan auf dem System installiert ist.	Der erwartete Zeichenfolgewert kann mehrere durch Kommas getrennte Versionen enthalten, z. B.: `9.2.0.8+` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`oracle.Client.Location`	Alle	Überprüft das Ausgangsverzeichnis des Oracle-Clients.	Zeichenfolge, z. B.: `/opt/oracle/products/10.1.0/client_1`
`oracle.Server`	Alle	Überprüft die Version des Oracle-Servers, der momentan auf dem System installiert ist.	Der erwartete Zeichenfolgewert kann mehrere durch Kommas getrennte Versionen enthalten, z. B.: `10.2.0.4g,11g R1` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`oracle.Server.Location`	Alle	Überprüft das Ausgangsverzeichnis des Oracle-Datenbankservers.	Zeichenfolge, z. B.: `/opt/oracle/product/10.1.0/Db_1`

Dateneigenschaften für Betriebssysteme

Die Dateneigenschaften für Betriebssysteme überprüfen die Voraussetzungen für das Betriebssystem, z. B. Version, Architektur, Gesamtspeicher, verfügbaren Hauptspeicher und physischen Gesamthauptspeicher. Für Windows-Systeme werden die VBScript-Collector für Betriebssysteme im Verzeichnis *ips_root*`/lib` mit der Präfix-ID `os` im Dateinamen verwendet. Für UNIX-Systeme werden die UNIX-Betriebssystemcollector im Verzeichnis *ips_root*`/UNIX_Linux` mit der Präfix-ID `os` im Dateinamen verwendet.

In Tabelle 24 sind die vorausgesetzten Eigenschaften für das Betriebssystem beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `os`.

Tabelle 24. Dateneigenschaften für Betriebssysteme

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.architecture`	Alle	Überprüft die Systemarchitektur.	`32-bit|64-bit`
`os.automount`	UNIX	Prüft, ob die Automount-Features funktionieren.	Boolescher Wert, z. B.: `True`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.autoUpdateEnabled`	Windows	Prüft, ob Windows-Updates automatisch aktiviert werden. Gibt `True` zurück, wenn ja.	Boolescher Wert, z. B.: `True`
`os.availableMemory`	Windows	Überprüft die virtuelle Hauptspeicherkapazität, die momentan verfügbar ist, aber vom Betriebssystem nicht genutzt wird.	Numerisches Format in MB, z. B.: `900MB`

`os.dir.`*`dir_name`*	UNIX	Überprüft das Dateisystem *dir_name* basierend auf den folgenden Qualifikationsattributen:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystems
		• Attribut `type` zum Bestimmen des zu prüfenden Dateisystemattributs, z. B. *`octal_digits`* für die Oktaldarstellung der Zugriffsberechtigungen für dieses Dateisystem
		dir_name kann beispielsweise Folgendes darstellen:
		• `tmp`
		• `home`
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[dir:`*`dir_name`*`, type:permission] `*`octal_digits`*`+`
			
			Verwenden Sie beispielsweise die folgende Zeichenfolge, um zu prüfen, ob das Ausgangsverzeichnis die Berechtigungen `drwxr-xr-x` hat:
			`os.dir.home=[dir:/home, type:permission]755+`

`os.diskquota`		Überprüft das Plattenbelegungskontingent für den angemeldeten Benutzer. Gibt den Wert für das Kontingent in KB oder den Wert `Unlimited` zurück.	Gültige Typen für den Wert: • Zahl für die Darstellung der Kilobyte, z. B. `414000` • Zeichenfolge für die Darstellung eines unbegrenzten Plattenkontingents, z. B. `Unlimited`
`os.expectLink`	UNIX	Prüft, ob die Expect-Erweiterung für TCL auf der Maschine verfügbar ist. Gibt `Available` zurück, wenn die Erweiterung verfügbar ist. **Anmerkung:** Die vorausgesetzte Eigenschaft `os.file.expect` prüft, ob die Expect-Erweiterung auf dem System installiert ist.	`Available|Unavailable`
`os.file.`*`script_name`*	UNIX	Prüft, ob das Script *script_name* auf der Maschine verfügbar ist. *script_name* kann beispielsweise für folgende Angaben stehen: • `bash` • `expect` • `gzip` • `tar`	Boolescher Wert, z. B.: `True`
`os.Firefox`	UNIX	Prüft, ob Mozilla Firefox auf der Maschine installiert ist. Gibt `Available` zurück, wenn ja.	`Available|Unavailable`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.FreePagingSpace`	UNIX	Überprüft die Gesamtgröße des verfügbaren Seitencaches.	Numerisches Format in MB oder GB, z. B.: `4GB+` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`os.ftpusers`	UNIX	Prüft, ob der Rootbenutzer in der Datei `ftpusers` aufgelistet ist, die die Benutzer bestimmt, denen keine FTP-Anmeldeberechtigungen erteilt werden. Gibt `Available` zurück, wenn der Benutzer nicht aufgelistet ist.	`Available|Unavailable`
`os.gnu.tar`	UNIX	Prüft, ob das GNU-Dienstprogramm "tar" auf der Maschine verfügbar ist. Gibt `Available` zurück, wenn das Dienstprogramm installiert ist.	`Available|Unavailable`
`os.hostformat`	UNIX	Überprüft, ob die Einträge in `/etc/host` das richtige Format haben.	Boolescher Wert, z. B.: `True`
`os.iodevicestatus`	AIX	Überprüft den Status der asynchronen Ein-/Ausgabe (`aio0`), d. h. den Kernelprozess für die Verbesserung der Leistung von E/A-Operationen. Gibt `Available` zurück, wenn der Kernelprozess verfügbar ist.	`Available|Unavailable`
`os.is8dot3File FormatEnabled`	Windows	Prüft, ob die 8.3-Dateinamensformate automatisch angewendet werden. Gibt `True` zurück, wenn diese angewendet werden.	Boolescher Wert, z. B.: `True`
`os.localhost InHostsFile`	Alle	Prüft, ob es einen Eintrag in der Datei "hosts" gibt, der den lokalen Host der Adresse `127.0.0.1` zuordnet, z. B.: `127.0.0.1 localhost`	Boolescher Wert, z. B.: `True`
`os.isService Running.`*`service_name`*	Windows	Verwenden Sie diese Namenskonvention, um zu prüfen, ob der Service *service_name* auf der Maschine aktiv ist. *service_name* kann beispielsweise Folgendes darstellen: • `remoteRegistry` für den Remoteregistrierungsdienst • `DNSClient` für den DNS-Clientdienst • `terminalServices` für Remotedesktopdienste oder Terminaldienste	Boolescher Wert, z. B.: `True`
`os.kernelMode`	AIX	Prüft, ob die CPU-Architektur den Kernelmodus oder den uneingeschränkten Modus unterstützt.	`32-bit|64-bit`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.kernelParameters`	Linux	Überprüft, ob die Kernelparameter für das Betriebssystem verfügbar sind.	`Available|Unavailable`
`os.kernelversion`	Linux	Überprüft das Release des Kernels für Linux-Betriebssysteme.	Numerisches Format, z. B. `2.6`
`os.largeFile`	UNIX	Prüft, ob große Dateien unterstützt werden.	Boolescher Wert, z. B.: `True`
`os.ldLibPath`	UNIX	Prüft, ob die Umgebungsvariable `LD_LIBRARY_PATH` vorhanden ist und mit einem Semikolon endet, d. h. `os.ldLibPath=[endsWith=:]`.	`Available|Unavailable`
`os.level`	AIX	Prüft, ob das Betriebssystem AIX einen höheren Stand als 10 für AIX Version 5.3 bzw. einen höheren Stand als 3 für AIX Version 6.1 hat.	Boolescher Wert, z. B.: `True`
`os.lib.`*`lib_name_version`*	UNIX	Überprüft, ob die unterstützte Version der Bibliothek *lib_name* auf dem System installiert ist. Zeichenfolge oder regulärer Ausdruck für die Darstellung von *lib_name_version*, z. B. in Fettschrift: • 32-Bit-Bibliothek **`libstdc++.so.#`** • 64-Bit-Bibliothek **`libstdc++.so.#`** • 32-Bit-Bibliothek **`libXft.so.#`** • 32-Bit-Bibliothek **`libXtst.so.#`** • 64-Bit-Bibliothek **`libaio.so.#`** • 32-Bit-XLC-Laufzeitversion **`xlC.rte`** • 32-Bit-Laufzeitumgebung **`xlC.aix50.rte`** für AIX Version 5.3 • 32-Bit-Laufzeitumgebung **`xlC.aix61.rte`** für AIX Version 6.1 • AIX-IOCP-Bibliothek **`bos.iocp.rte`** • **`bos.loc.iso.en_us`**, die ISO-Codedateigruppe für das AIX-Basisbetriebssystem	Gültige Typen für den Wert: Zeichenfolge, z. B.: • `/usr/lib/libstdc++.so.#` als Wert für die 32-Bit-Bibliothek `libstdc++.so.#` • `/usr/lib64/libaio.so.#` als Wert für die 64-Bit-Bibliothek `libaio.so.#` • `xlC.aix50.rte.9.0.0.8+` als Wert für die 32-Bit-XLC-Laufzeitumgebung `xlC.aix50.rte` für AIX Version 5.3 • `bos.loc.iso.en_us` für die ISO-Code-Dateigruppe `regex {`*`str`*`}`, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für den Bibliotheksnamen darstellt, z. B.: `regex{.*libgcc.*}` Prüft, ob eine Version der GCC-Low-Level-Laufzeitbibliothek `libgcc`für dieses Betriebssystem existiert. **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
os.loginVariable	UNIX	Überprüft, ob die Standardpfade für den Rootbenutzer in den Variablen PATH und SUPATH definiert sind. Gibt Available zurück, wenn sie gesetzt sind.	Available|Unavailable
os.maximoDirectory	UNIX	Prüft, ob das Verzeichnis /export/home/maximo verfügbar ist.	Available|Unavailable
os.maximoDirOwner	UNIX	Überprüft den Eigner des Verzeichnisses /export/home/maximo.	maximo
os.maximumProcesses	UNIX	Überprüft die maximale Anzahl an Prozessen, die für jeden Benutzer ausgeführt werden können.	Zahl, z. B. 2048
os.MozillaVersion	UNIX	Prüft, ob eine bestimmte Version von Mozilla Firefox auf der Maschine ist (anders als die vorausgesetzte Eigenschaft os.Firefox).	Numerisches Format, z. B. 3.0+ **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die im Abschnitt Tabelle 1 auf Seite 2 beschrieben sind.
os.mountcheck	UNIX	Prüft basierend auf den folgenden Qualifikationsattributen, ob das Dateisystem angehängt ist: • Attribut drive zum Bestimmen, welches Verzeichnis das angehängte Dateisystem ist • Attribut nosuid zum Bestimmen, ob die Mountoption beim Anhängen des Dateisystems gesetzt ist	Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat: [Laufwerk:*dir_name*, *mount_option*: false|true] True|False Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Verzeichnis /home angehängt ist und die Option nosuid nicht gesetzt ist: os.mountcheck=[drive:/home, nosuid:false]True

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.package.`*`package_name`*	UNIX	Prüft, ob die unterstützte Version des Pakets *package_name* auf der Maschine installiert ist. Zeichenfolge für die Darstellung von *package_name*, z. B. in Fettschrift: • **`bash`** für die Shell • **`expect`** für das TCL-Erweiterungspaket • **`libgcc`** für das GCC-Low-Level-Laufzeitpaket • **`openssh`** für die Open-Source-Secure-Shell • **`openssl`** für das Open-Source-Toolkit für SSL/TLS • **`perl`** für das Perl-Scripting-Paket • **`rpm`** für RPM- oder RPM-Build-Pakete • **`telnet`** für das Telnet-Paket • **`wget`** für das GNU-Paket für Dateiabruf	Zeichenfolge, z. B.: • `bash-3.2+` for bash shell • `expect-1.2.0` für Expect • `libgcc-3.4.3-9` für libgcc • `openssh-5.0.0.5301-` für openssh • `openssl-4.2.0-` für OpenSSL • `perl-5.8.2` für Perl • `rpm` • `telnet` • `wget` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`os.pagesize`	UNIX	Prüft die Seitengröße des Systems.	Numerisches Format in KB, z. B.: `4KB` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`os.RAMSize`	Alle	Prüft den Arbeitsspeicher des Systems.	Numerisches Format in GB, z. B. `8GB`
`os.SeaMonkeyVersion`	UNIX	Prüft, ob eine bestimmte Version von Mozilla SeaMonkey auf dem Computer unter dem mit der Umgebungsvariablen `PATH` gesetzten Pfad vorhanden ist.	Numerisches Format, z. B. `2.10` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte

`os.SELinux`	Linux	Überprüft den Aktivierungsstatus des Linux-Features Security-Enhancement basierend auf den folgenden Qualifikationsattributen:
		• Attribut `source` zum Bestimmen des für das relevante Betriebssystem zu verwendenden Befehls
			Gültige Typen für den Wert:
			• Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[source:Command] Disabled|Enabled`
			Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Feature inaktiviert ist oder einen Berechtigungsstatus unter dem Betriebssystem Red Hat oder SUSE hat:
			`os.SELinux=[source: Command]Disabled`
			• Zeichenfolge ohne Qualifikationsmerkmal, in der das Betriebssystem eine generische Linux-Variante ist:
			`os.SELinux=Disabled`

`os.servicePack`	Alle	Prüft die aktuelle Version des installierten Service-Packs.	Numerisches Format mit *majorVersion*. *minorVersion* oder nur *majorVersion* Beispielsweise kann mit der folgenden Angabe geprüft werden, ob Service-Pack 2 oder höher installiert ist: `2+` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die im Abschnitt Tabelle 1 auf Seite 2 beschrieben sind.
`os.shell.default`	UNIX	Prüft, ob das Standard-Shell-Script installiert ist.	Zeichenfolge für die Darstellung des Shell-Script, z. B. `bash`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
os.space.*dir_name* (vorausgesetzte Eigenschaften) Prerequisite Scanner hat drei Varianten der Eigenschaft os.space.*dir_name*: • os.space.*dir_name* prüft, ob genügend Plattenspeicherplatz für das angegebene Dateisystem verfügbar ist, unabhängig davon, ob der angemeldete Benutzer immer der Rootbenutzer oder immer ein Benutzer ohne Rootberechtigung ist. Verwenden Sie diese Variante der vorausgesetzten Eigenschaft, wenn Sie den angegebenen Pfad des Dateisystems überprüfen möchten, aber es keine Rolle spielt, ob der angemeldete Benutzer immer der Rootbenutzer oder immer ein Benutzer ohne Rootberechtigung ist. **Anmerkung:** Sie können diese Variante nicht zweimal für dasselbe Dateisystem, aber für andere Benutzertypen in einer Konfigurationsdatei verwenden. Verwenden Sie stattdessen eine Kombination der beiden anderen Varianten. • os.space.*dir_name*_nonroot prüft, ob genügend Plattenspeicherplatz für das angegebene Dateisystem des Benutzers ohne Rootberechtigung verfügbar ist. Verwenden Sie diese Variante der vorausgesetzten Eigenschaft, wenn Sie als Benutzer ohne Rootberechtigung angemeldet sind und explizit den angegebenen Pfad für das Dateisystem überprüfen möchten. **Anmerkung:** Der Benutzer ohne Rootberechtigung muss derselbe Benutzer sein, der auch das Produkt auf dem Zielsystem installiert. • os.space.*dir_name*_root prüft, ob genügend Plattenspeicherplatz für das angegebene Dateisystem des Rootbenutzers verfügbar ist. Verwenden Sie diese Variante der vorausgesetzten Eigenschaft, wenn Sie als Rootbenutzer angemeldet sind und explizit den angegebenen Pfad für das Dateisystem überprüfen möchten. Sie können die Varianten os.space.*dir_name*_nonroot und os.space.*dir_name*_root in derselben Konfigurationsdatei angeben. Prerequisite Scanner gibt NOT_REQ_CHECK_ID in der Zelle mit den tatsächlichen Ergebnissen für die nicht zutreffende Variante aus. Wenn der anmeldete Benutzer beispielsweise der Rootbenutzer ist, gibt Prerequisite Scanner NOT_REQ_CHECK_ID für die Variante os.space.*dir_name*_nonroot aus.			

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte

os.space.*dir_name*	UNIX	Überprüft den verfügbaren Plattenspeicherplatz für das angegebene Dateisystem <*dir_name*> basierend auf einem oder mehreren der folgenden Qualifikationsattribute:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystempfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
		
		Der Wert für das Attribut `dir` hängt vom angemeldeten Benutzer ab. Deshalb ist der Wert ein Name/Wert-Paar für die Darstellung des Benutzertyps, z. B. Rootbenutzer oder Benutzer ohne Rootberechtigungen, und des zugehörigen Pfads.
		
		dir_name kann beispielsweise Folgendes darstellen:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
		
		Anmerkung: Sie können diese Variante nicht zweimal für dasselbe Dateisystem, aber für andere Benutzertypen in einer Konfigurationsdatei verwenden. Verwenden Sie eine Kombination der Varianten `os.space.`*`dir_name`*`_nonroot` und `os.space.`*`dir_name`*`_root`.
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Rootbenutzers:
			`[dir:`**`root`**`=<`*`dir_path`*`>, unit:<`*`unit_name`*`>] <`*`disk_space`*`>`
			
			Beispiel:
			`os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200`
			
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Benutzers ohne Rootberechtigung:
			`[dir:`**`non_root`**`=<`*`dir_path`*`>, unit:<`*`unit_name`*`>] <disk_space>`
			
			Beispiel:
			`os.space.home= [dir:non_root=USERHOME/.acsi_HOST, unit:MB]200`
			
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat mit nur einem einzigen Qualifikationsmerkmal:
			`[dir:<`*`dir_path`*`>] <`*`disk_space`*`> MB`
			
			Beispiel:
			`os.space.home= [dir:/home/sat]250MB`
			
			Numerisches Format in MB oder GBs, z. B.:
			`os.space.opt=11GB`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte

os.space.*dir_name*_nonroot	UNIX	Überprüft den verfügbaren Plattenspeicherplatz für das Dateisystem *dir_name* des Benutzers ohne Rootberechtigung basierend auf einem oder mehreren der folgenden Qualifikationsattribute:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystempfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
		dir_name kann beispielsweise Folgendes darstellen:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Benutzers ohne Rootberechtigung:
			`[dir:`**`non_root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] disk_space`
			Beispiel:
			`os.space.home_nonroot= [dir:non_root= USERHOME/.acsi_HOST, unit:MB]200`
			Zeichenfolge mit dem Qualifikationsattribut `dir` für das Dateisystem eines Benutzers ohne Rootberechtigung:
			`[dir:`**`non_root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			Beispiel:
			`os.space.opt_nonroot= [dir:non_root=/opt/IBM/ITM] 1024MB`

os.space.*dir_name*_root	UNIX	Überprüft den verfügbaren Plattenspeicherplatz für das Dateisystem *dir_name* des Rootbenutzers basierend auf einem oder mehreren der folgenden Qualifikationsattribute:
		• Attribut `dir` zum Bestimmen des zu prüfenden Dateisystempfads
		• Attribut `unit` zum Bestimmen der für den Plattenspeicherplatz zu verwendenden Einheiten
		dir_name kann beispielsweise Folgendes darstellen:
		• `home`
		• `opt`
		• `tmp`
		• `usr`
		• `var`
			Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Rootbenutzers:
			`[dir:`**`root`**`=`*`dir_path`*`, unit:`*`unit_name`*`] `*`disk_space`*
			Beispiel:
			`os.space.usr_root= [dir:root= /usr/ibm/common/acsi, unit:GB]200`
			Zeichenfolge mit dem Qualifikationsattribut `dir` für das Dateisystem eines Rootbenutzers:
			`[dir:`**`root`**`=`*`dir_path`*`] `*`disk_space`*`GB|MB`
			Beispiel:
			`os.space.opt_root= [dir:root=/opt/IBM/ITM] 1024MB`

`os.sshdConfig`	UNIX	Prüft, ob eine zulässige Rootanmeldung für SSH-Dämonsitzungen konfiguriert ist.	`Available|Unavailable`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.swapSize`	UNIX	Prüft, ob der Auslagerungsspeicher größer als der Arbeitsspeicher oder die Gesamtauslagerungsspeicherkapazität ist.	Gültige Typen für den Wert: • Boolescher Wert, z. B.: `True` • Numerisches Format in MB oder GB, z. B.: `2GB`

`os.tmpdir`	UNIX	Überprüft die Zugriffsberechtigungen, die dem Dateisystem `/tmp` zugeordnet sind, einschließlich aller speziellen Berechtigungen, die mit Flags für Zugriffsrechte gesetzt werden, z. B. `sticky`-, `setuid`- oder `setgid`-Bit in Oktalziffern.	Zahl für die Darstellung der Oktalstellen `octal_digits` für die Zugriffsberechtigungen.
			Geben Sie beispielsweise Folgendes an, um zu prüfen, ob das temporäre Verzeichnis die Berechtigungen `drwxrwxrwt` mit aktiviertem Sticky Bit hat:
			`1777`
			Geben Sie beispielsweise Folgendes an, um zu prüfen, ob das temporäre Verzeichnis die Berechtigungen `drwxrwxrwx` ohne das Sticky Bit hat:
			`777`

`os.totalMemory`	Windows	Die Gesamtgröße des virtuellen Speichers, auf den das Betriebssystem zugreifen kann.	Numerisches Format in MB oder GB, z. B. `4GB`
`os.totalPhysicalMemory`	Windows	Die Gesamtgröße des physischen Speichers, auf den das Betriebssystem zugreifen kann. Gibt aber nicht den tatsächlichen physischen Speicher auf dem Zielcomputer an.	Numerisches Format in MB oder GB, z. B. `2030MB`

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte

`os.ulimit`	UNIX	Prüft basierend auf den folgenden Qualifikationsattributen, ob eine unbegrenzte Zahl an Prozessen ausgeführt werden kann:
		• Attribut `type` zur Bestimmung des zusätzlich zu prüfenden Grenzwerts. `filedescriptorlimit` überprüft beispielsweise den Grenzwert für die Anzahl der Dateideskriptoren, die Prozesse öffnen können.
		
		Alternativ prüft diese Eigenschaft, ob die folgenden Grenzwerte für die angegebenen Domänen in der Datei `/etc/security/limits.conf` gesetzt wurden:
		`root - stack unlimited`
		`ctginst1 - stack unlimited`
		`root - nofile 8192`
		`tioadmin - nofile 32767`
			Gültige Typen für den Wert:
			• Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
			`[type:`*`limit_name`*`]`
			`limit_value``,`
			`limited|unlimited`
			Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob der Grenzwert für Dateideskriptoren höher als 8192 mit einer unbegrenzten Prozessanzahl ist:
			`os.ulimit=[type:`
			`filedescriptorlimit]`
			`8192+,`
			`unlimited`
			
			Im Folgenden sind die gültige Typen für die Grenzwertüberprüfung aufgelistet, wobei *limit_name* den Typ des Grenzwerts darstellt:
			– `ALL` (überprüft alle Grenzwerte)
			– `corefilesizelimit`
			– `datasegmentlimit`
			– `filedescriptorlimit`
			– `filesizelimit`
			– `hardlimit`
			– `processlimit`
			– `maxmemorysizelimit`
			– `maxprocesseslimit`
			– `stacksizelimit`
			– `threadlimit`
			• `Available|Unavailable` gibt an, ob für die relevanten Domänen Grenzwerte in der Datei `/etc/security/limits.conf` gesetzt wurden.

`os.umask`	UNIX	Überprüft die Berechtigungen für die Erstellungsmaske für den Dateimodus.	Zahl für die Darstellung der Oktalziffern *`octal_digits`* für die Zugriffsberechtigungen. Um beispielsweise zu prüfen, ob die neuen Dateien nur für den Eigner beschreibbar sind, setzen Sie die Oktalziffer auf `0022`.

Tabelle 24. Dateneigenschaften für Betriebssysteme (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`os.userLimits`	UNIX	Prüft, ob die maximale Stapelgröße unbegrenzt ist. Gibt `Available` zurück, wenn die Stapelgröße unbegrenzt ist.	`Available|Unavailable`
`os.versionNumber`	Windows	Überprüft die aktuelle Version des Betriebssystems, das auf der Maschine installiert ist.	Numerisches Format, z. B. `5.0+` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die im Abschnitt Tabelle 1 auf Seite 2 beschrieben sind.
`os.windowManager`	UNIX	Prüft, ob GNOME oder KDE als grafischer Desktop verfügbar ist.	`Available|Unavailable`

Dateneigenschaften für installierte Software

Die Dateneigenschaften für installierte Software überprüfen die Voraussetzungen für installierte Software, wie z. B. die in der Windows-Registry registrierten Programme. Außerdem wird geprüft, ob cygwin und gskit installiert sind. Für Windows-Systeme werden die Collector für installierte Software aus dem Verzeichnis *IPS-Stammverzeichnis*`/lib` mit der Präfix-ID `installedSoftware`, `cygwin`, oder `gskit` im Dateinamen verwendet.

In Tabelle 25 sind die allgemeinen vorausgesetzten Dateneigenschaften beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert keine Präfix-ID.

Tabelle 25. Dateneigenschaften für installierte Software

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`installedSoftware`	Windows	Scannt die Betriebssystemregistry nach installierten Programmen mit Positionen.	Zeichenfolge mit mehreren durch Kommas getrennten Anwendungen.
`cygwinVersion`	Windows	Überprüft die Version von cygwin, die auf der Maschine installiert ist. Gibt 0.0 zurück, wenn keine Version installiert ist.	Positive ganze Zahl, z. B. `1.5` **Anmerkung:** In den Werten können die Sonderzeichen verwendet werden, die in Tabelle 1 auf Seite 2 beschrieben sind.
`gskit7Version`	Windows	Prüft, ob gskit Version 7 auf der Maschine installiert ist. Gibt 0.0 zurück, wenn Version 7 nicht installiert ist.	Positive ganze Zahl, z. B. `7.0`
`gskit8Version`	Windows	Prüft, ob gskit Version 8 auf der Maschine installiert ist. Gibt 0.0 zurück, wenn Version 8 nicht installiert ist.	Positive ganze Zahl, z. B. `8.0`

Dateneigenschaften für Benutzer

Die Dateneigenschaften für Benutzer überprüfen Voraussetzungen für Benutzer, z. B., ob der angemeldete Benutzer Administratorberechtigungen hat oder ob er der Rootbenutzer ist. Für Windows-Systeme wird der Benutzercollector im Verzeichnis *IPS-Stammverzeichnis*/lib mit der Präfix-ID user im Dateinamen verwendet. Für UNIX-Systeme wird der Benutzercollector in *IPS-Stammverzeichnis*/lib/packageTest.sh verwendet.

In Tabelle 26 sind die vorausgesetzten Eigenschaften für Benutzer beschrieben. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID user.

Tabelle 26. Dateneigenschaften für Benutzer

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
user.userID	Windows	Die ID des momentan angemeldeten Benutzers.	Zeichenfolge, z. B. smithj
user.isAdmin	Alle	Prüft, ob der angemeldete Benutzer ein Mitglied der Administratorgruppe ist.	Boolescher Wert, z. B. True

Dateneigenschaften für Windows-Netze

Die Dateneigenschaften für Windows-Netze überprüfen Netzvoraussetzungen, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften. Es werden die Windows-Netzcollector im Verzeichnis *IPS-Stammverzeichnis*/lib mit der Präfix-ID network im Dateinamen verwendet.

In Tabelle 27 sind die vorausgesetzten Eigenschaften für Netze beschrieben, die für alle Windows-Plattformen gelten. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID network.

Tabelle 27. Dateneigenschaften für Windows-Netze

Vorausgesetzte Eigenschaft	Beschreibung	Gültige Werte
network.DHCPEnabled	Prüft, ob mindestens ein Adapter mit einer gültigen IP-Adresse diese IP-Adresse mit DHCP angefordert hat. Gibt True zurück, wenn ja.	Boolescher Wert, z. B. False
network.netBIOSEnabled	Prüft, ob mindestens ein Adapter mit einer gültigen IP-Adresse NetBIOS als aktiviertes Protokoll hat. Gibt True zurück, wenn ja.	Boolescher Wert, z. B. True
network.pingLocalhost	Prüft, ob der lokale Host auf das Pingprotokoll antwortet. Gibt True zurück, wenn ja.	Boolescher Wert, z. B. True
network.pingSelf	Prüft, ob der Name des lokalen Computers mit DCHP aufgelöst wurde ob der Computer mit Ping erreichbar ist. Gibt True zurück, wenn ja.	Boolescher Wert, z. B. True
network.ValidateHostsFile	Prüft, ob die Einträge in der Datei C:\WINDOWS\system32\drivers\etc\hosts das richtige Format haben. Gibt True zurück, wenn das Format gültig ist.	Boolescher Wert, z. B. True

Dateneigenschaften für UNIX-Netze

Die Dateneigenschaften für UNIX-Netze überprüfen Netzvoraussetzungen, z. B., ob NetBIOS und DHCP auf der Maschine aktiviert sind, sowie Ping-Eigenschaften. Es werden die Netzcollector im Verzeichnis *IPS-Stammverzeichnis*/UNIX_Linux verwendet.

In Tabelle 28 sind die vorausgesetzten Eigenschaften für Netze beschrieben, die für alle UNIX-Plattformen gelten. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `network`.

Tabelle 28. Dateneigenschaften für UNIX-Netze

Vorausgesetzte Eigenschaft	Beschreibung	Gültige Werte
`network.DHCPEnabled`	Prüft, ob mindestens ein Adapter mit einer gültigen IP-Adresse diese IP-Adresse mit DHCP angefordert hat.	Boolescher Wert, z. B. `False`
`network.dns`	Prüft, ob der DNS-Eintrag für die Hostmaschine korrekt ist.	Boolescher Wert, z. B. `True`
`network.fqdn`	Überprüft, ob der vollständig qualifizierte Domänenname für die Hostmaschine definiert ist.	Boolescher Wert, z. B. `True`
`network.pingLocalhost`	Prüft, ob der lokale Host auf das Pingprotokoll antwortet.	Boolescher Wert, z. B. `True`
`network.pingSelf`	Prüft, ob der Name des lokalen Computers mit DCHP aufgelöst wurde ob der Computer mit Ping erreichbar ist.	Boolescher Wert, z. B. `True`

Dateneigenschaften für Umgebungsvariablen

Die Dateneigenschaften für Umgebungsvariablen überprüfen die Voraussetzungen für Umgebungsvariablen, die für alle Plattformen gelten können. Sie prüfen beispielsweise, ob eine Umgebungsvariable gesetzt ist, oder sie überprüfen den Wert einer Umgebungsvariablen. Für Windows-Systeme werden die Umgebungsvariablencollector im Verzeichnis *IPS-Stammverzeichnis*`/lib` mit der Präfix-ID `env` im Dateinamen verwendet. Für UNIX-Systeme werden die UNIX-Umgebungsvariablencollector im Verzeichnis *IPS-Stammverzeichnis*`/UNIX_Linux` mit der Präfix-ID `env` im Dateinamen verwendet.

In Tabelle 29 sind die vorausgesetzten Eigenschaften für Umgebungsvariablen beschrieben, die für alle Plattformen gelten. Diese Kategorie vorausgesetzter Eigenschaften erfordert die Präfix-ID `env`.

Tabelle 29. Dateneigenschaften für Umgebungsvariablen

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`env.var.set.`*`env_var_name`*	UNIX	Verwenden Sie diese Namenskonvention, um zu prüfen, ob die mit *env_var_name* angegebene Umgebungsvariable auf dem Computer gesetzt ist, z. B.: • `env.var.set.HOME` prüft, ob die Umgebungsvariable für das Ausgangsverzeichnis gesetzt ist, wobei *env_var_name* für den Namen der Umgebungsvariablen `HOME` steht. • `env.var.set.JAVA_HOME` prüft, ob die Umgebungsvariable für das Ausgangsverzeichnis für Java gesetzt ist, wobei *env_var_name* für den Namen der Umgebungsvariablen `JAVA_HOME` steht.	Boolescher Wert, z. B. `True`

Tabelle 29. Dateneigenschaften für Umgebungsvariablen (Forts.)

Vorausgesetzte Eigenschaft	Plattform	Beschreibung	Gültige Werte
`env.var.set.`*`env_var_name`* [`type:`*`env_var_type`*]	Windows	Verwenden Sie diese Namenskonvention, um zu prüfen, ob die mit *env_var_name* angegebene Umgebungsvariable für den mit *env_var_type* angegebenen Umgebungsvariablentyp gesetzt ist, z. B.: • `env.var.set.HOME` prüft, ob die Umgebungsvariable für das Ausgangsverzeichnis gesetzt ist, wobei *env_var_name* für den Namen der Umgebungsvariablen `HOME` steht. • `env.var.set.JAVA_HOME[type:User]` prüft, ob die Umgebungsvariable für das Java-Ausgangsverzeichnis für den angemeldeten Benutzer gesetzt ist, wobei *env_var_name* für den Namen der Umgebungsvariablen `JAVA_HOME` und *env_var_type* für den Typ (`User`) der Umgebungsvariablen steht. Der Umgebungsvariablentyp *env_var_type* ist optional und stellt die Typen von Umgebungsvariablen dar, die vom Windows-Betriebssystem unterstützt werden: • `Process` • `System` • `User` • `Volatile` Der Standardtyp ist `Process`.	Boolescher Wert, z. B. `True`
`env.classpath.derbyJAR`	Alle	Prüft, ob der Pfad zur Derby-JAR-Datei in der Umgebungsvariablen für den Klassenpfad enthalten ist.	Boolescher Wert, z. B. `True`
`env.CIT.homeExists`	Windows	Prüft, ob die Umgebungsvariablen `HOMEDRIVE` und `HOMEPATH` vorhanden sind.	Boolescher Wert, z. B. `True`

Anhang D. Vordefinierte Collector für UNIX-Systeme

Es gibt einzelne Collectors für Prüfungen vorausgesetzter Eigenschaften auf UNIX-Systemen, die im Verzeichnis *ips_root*/lib gespeichert sind. Sie können sich diese Collector und deren Eingabeparameter ansehen, bevor Sie angepasste Collector erstellen.

In Tabelle 30 sind die vordefinierten Collector für UNIX-Systeme beschrieben.

Tabelle 30. UNIX-Collector

Collector	Für vorausgesetzte Eigenschaft	Eingaben
DB2_Version	DB2 Version	Ohne
DBType	DBType	Ohne
DBTypeDetails	DBTypeDetails	Ohne
env.classpath.derbyJAR	env.classpath.derbyJAR	Ohne
env.var.set	env.var.set*env_var_name* *env_var_name* ist der Name der zu prüfenden Umgebungsvariablen.	*$env_var_name*
network.DHCPEnabled	network.DHCPEnabled	Ohne
network.dns	network.dns	Ohne
network.fqdn	network.fqdn	Ohne
network.pingSelf	network.pingSelf	Ohne
network.port	network.availablePorts.* network.portsInUse.*	$ports
oracle.Client	oracle.Client	Ohne
oracle.Client.Location	oracle.Client.Location	Ohne
oracle.Server	oracle.Server	Ohne
oracle.Server.Location	oracle.Server.Location	Ohne
os.architecture	os.architecture	32 bit|64 bit
os.automount	os.automount	Ohne
os.cmd	os.lookup	nslookup
os.cmd	os.tar|os.gnu.tar	tar|gtar
os.dir	os.dir.*dir_name* *dir_name* kann beispielsweise für folgende Angaben stehen: • tmp • home	Zeichenfolge im folgenden Format: [dir:*dir_name*, type:permission] *octal_digits*+ Die folgende Zeichenfolge prüft beispielsweise, ob das Verzeichnis *dir_name*, d. h. das Ausgangsverzeichnis, die Berechtigungen drwxr-xr-x hat: [dir:/home, type:permission]755+
os.diskquota	os.diskquota	Ohne

Tabelle 30. UNIX-Collector (Forts.)

Collector	Für vorausgesetzte Eigenschaft	Eingaben
os.expectLink	os.expectLink	Ohne
os.filepath	os.file.*script_name* *script_name* kann beispielsweise für folgende Angaben stehen: • bash • expect • gzip • tar	Pfad zur Scriptdatei, wie z. B.: • /bin/bash • /usr/bin/expect • /usr/bin/gzip • /usr/bin/tar
os.Firefox	os.Firefox	Ohne
os.FreePagingSpace	os.FreePagingSpace	Ohne
os.ftpusers	os.ftpusers	Ohne
os.hostformat	os.hostformat	Ohne
os.iodevicestatus	os.iodevicestatus	Ohne
os.kernelMode	os.kernelMode	Ohne
os.kernelParameters	os.kernelParameters	Ohne
os.kernelversion	os.kernelversion	Ohne
os.largeFile	os.largeFile	Ohne
os.ldLibPath	os.ldLibPath	Produkt
os.level	os.level	Ohne

Tabelle 30. UNIX-Collector (Forts.)

Collector	Für vorausgesetzte Eigenschaft	Eingaben
os.lib	os.lib.*lib_name_version* Zeichenfolge oder regulärer Ausdruck für die Darstellung von *lib_name_version*, z. B. in Fettschrift: • 32-Bit-Bibliothek **libstdc++.so.#** • 64-Bit-Bibliothek **libstdc++.so.#** • 32-Bit-Bibliothek **libXft.so.#** • 32-Bit-Bibliothek **libXtst.so.#** • 64-Bit-Bibliothek **libaio.so.#** • 32-Bit-XLC-Laufzeitversion **xlC.rte** • 32-Bit-Laufzeitumgebung **xlC.aix50.rte** für AIX Version 5.3 • 32-Bit-Laufzeitumgebung **xlC.aix61.rte** für AIX Version 6.1 • AIX-IOCP-Bibliothek **bos.iocp.rte** • **bos.loc.iso.en_us**, die ISO-Codedateigruppe für das AIX-Basisbetriebssystem • **regex{*str*}**, ein regulärer Ausdruck mit dem Eingabeparameter *str*, der das Suchmuster für den Bibliotheksnamen darstellt, z. B. .*libgcc.*	*path_to_library* oder *lib_name_version* Mit den Eingabeparametern /usr/lib/libstdc++.so.5 und libstdc++.so: wird beispielsweise der tatsächliche Wert für die vorausgesetzte Eigenschaft os.lib.libstdc++.so gesucht: os.lib /usr/lib/libstdc++.so.5 libstdc++.so Mit dem folgenden Eingabeparameter wird beispielsweise geprüft, ob eine Version der GCC-Low-Level-Laufzeitbibliothek libgcc auf der Maschine vorhanden ist: regex{.*libgcc.*}
os.loginVariable	os.loginVariable	Ohne
os.maximoDirectory	os.maximoDirectory	Ohne
os.maximoDirOwner	os.maximoDirOwner	Ohne
os.maximumProcesses	os.maximumProcesses	Ohne
os.MozillaVersion	os.MozillaVersion	Ohne
os.mountcheck	os.mountcheck	Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat: [Laufwerk:*dir_name*, *mount_option*: false|true] True|False Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Verzeichnis /home angehängt ist und die Option nosuid nicht gesetzt ist: os.mountcheck=[drive:/home, nosuid:false]True

Tabelle 30. UNIX-Collector (Forts.)

Collector	Für vorausgesetzte Eigenschaft	Eingaben
`os.package`	`os.package.`*`package_name`* Zeichenfolge für die Darstellung von *package_name*, z. B. in Fettschrift: • **`bash`** für die Shell • **`expect`** für das TCL-Erweiterungspaket • **`libgcc`** für das GCC-Low-Level-Laufzeitpaket • **`openssh`** für die Open-Source-Secure-Shell • **`openssl`** für das Open-Source-Toolkit für SSL/TLS • **`perl`** für das Perl-Scripting-Paket • **`rpm`** für RPM- oder RPM-Build-Pakete • **`telnet`** für das Telnet-Paket • **`wget`** für das GNU-Paket für Dateiabruf	*package_name*, wobei *package_name* gleich `rpm` ist: `os.package rpm`
`os.pagesize`	os.pagesize	Ohne
`os.RAMSize`	os.RAMSize	Ohne

`os.SELinux`	`os.SELinux`	• Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat:
		`[source:Command] Disabled|Enabled`
		Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob das Feature inaktiviert ist oder einen Berechtigungsstatus unter dem Betriebssystem Red Hat oder SUSE hat:
		`os.SELinux=[source: Command]Disabled`
		• Wenn kein Qualifikationsmerkmal vorhanden ist, wird kein Wert an den Collector übergeben.

`os.servicePack`	`os.servicePack`	Service-Pack-Wert
`os.shell.default`	`os.shell.default`	Der erwartete Wert für die vorausgesetzte Eigenschaft, z. B. `bash`

Tabelle 30. UNIX-Collector (Forts.)

Collector	Für vorausgesetzte Eigenschaft	Eingaben
os.space	os.space.*dir_name* *dir_name* kann beispielsweise für folgende Angaben stehen: • usr • home • tmp • var	Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Rootbenutzers: [dir:**root**=*dir_path*, unit:*unit_name*] *disk_space* Beispiel: os.space.usr= [dir:root=/usr/ibm/common/acsi, unit:GB]200 Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat für das Dateisystem eines Benutzers ohne Rootberechtigung: [dir:**non_root**=*dir_path*, unit:*unit_name*] disk_space Beispiel: os.space.home= [dir:non_root=USERHOME/ .acsi_HOST, unit:MB]200 Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat mit nur einem einzigen Qualifikationsmerkmal: [dir:*dir_path*] *disk_space* MB Beispiel: os.space.home= [dir:/home/sat]250MB
os.sshdConfig	os.sshdConfig	Ohne
os.swapSize	os.swapSize	Ohne
os.tmpdir	os.tmpdir	Ohne

Tabelle 30. UNIX-Collector (Forts.)

Collector	Für vorausgesetzte Eigenschaft	Eingaben
os.ulimit	os.ulimit	Zeichenfolge mit dem folgenden Qualifikationsmerkmalformat: [type:*limit_name*] *limit_value*, limited|unlimited Mit der folgenden Zeichenfolge wird beispielsweise geprüft, ob der Grenzwert für Dateideskriptoren höher als 8192 mit einer unbegrenzten Prozessanzahl ist: os.ulimit= [type:filedescriptorlimit] 8192+,unlimited Im Folgenden sind die gültige Typen für die Grenzwertüberprüfung aufgelistet, wobei *limit_name* den Typ des Grenzwerts darstellt: • ALL (überprüft alle Grenzwerte) • corefilesizelimit • datasegmentlimit • filedescriptorlimit • filesizelimit • hardlimit • processlimit • maxmemorysizelimit • maxprocesseslimit • stacksizelimit • threadlimit
os.umask	os.umask	Ohne
os.userLimits	os.userLimits	Ohne
os.windowManager	os.windowManager	Ohne

Anhang E. Allgemeine Funktionen für Windows-Systeme

Prerequisite Scanner enthält eine Gruppe allgemeiner Funktionen in der Datei `/lib/common_function.vbs` für die Durchführung von Prüfungen auf Windows-Systemen.

Tabelle 31. Funktionen in common_function.vbs

Funktion	Beschreibung
„allFiles()" auf Seite 124	Liest die Dateinamen in einem angegebenen Verzeichnis in ein Array ein.
„arrayToString()" auf Seite 125	Erstellt eine Zeichenfolgedarstellung für das Array.
„bigthan()" auf Seite 125	Berechnet die Differenz zwischen dem erwarteten und dem tatsächlichen Wert der vorausgesetzten Eigenschaft, wenn diese vorausgesetzte Eigenschaft in MB oder GB angegeben ist.
„changeMG()" auf Seite 126	Konvertiert den Eingabeparameter für die erforderlichen Eigenschaften für den Plattenspeicherplatz und den Hauptspeicher in MB oder GB.
„checkItemToString()" auf Seite 126	Erstellt eine Zeichenfolgedarstellung für das Objekt `CheckItem`.
„dictionaryToString()" auf Seite 127	Erstellt eine Zeichenfolgedarstellung für das Scripting-Wörterbuchobjekt.
„exeCommand()" auf Seite 127	Führt den angegebenen Befehl aus und gibt das Ergebnis dieser Befehlsausführung zurück.
„filterCommand()" auf Seite 128	Führt den angegebenen Befehl aus und gibt die Zeilen aus dem Ergebnis des Befehls zurück, die mit dem angegebenen Muster übereinstimmen.
„filterFile()" auf Seite 128	Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und filtert ihn.
„findNewest()" auf Seite 129	Sucht die neueste Konfigurationsdatei.
„findSuitableFile()" auf Seite 129	Sucht die relevante Konfigurationsdatei für ein Produkt und eine Version.
„fmt()" auf Seite 130	Ändert eine Zeichenfolge, indem der Zeichenfolge eine bestimmte Anzahl an Zeichen aus einer anderen Zeichenfolge hinzugefügt und die andere Zeichenfolge mit Leerzeichen aufgefüllt wird, wenn die andere Zeichenfolge zu kurz ist, bzw. die andere Zeichenfolge gekürzt wird, wenn sie zu lang ist.
„formatForDisplay()" auf Seite 131	Formatiert den Eingabeparameter, um ihn lesbar machen.

Tabelle 31. Funktionen in common_function.vbs (Forts.)

Funktion	Beschreibung
„formatSizeForDisplay()" auf Seite 131	Akzeptiert den Eingabeparameter und fügt dem Eingabeparameter zwei Nachkommastellen hinzu bzw. kürzt die Nachkommastellen des Eingabeparameters auf zwei Dezimalstellen. Aus 123MB wird beispielsweise 123,00MB und aus 12,123MBs wird 12,12MBs.
„getDecimalSeparator()" auf Seite 132	Bestimmt das Dezimaltrennzeichen, das für die aktuelle Ländereinstellung verwendet wird.
„getFirstMatch()" auf Seite 132	Ruft die erste Übereinstimmung für den Suchbegriff im Array ab.
„isMatch()" auf Seite 133	Prüft, ob das Suchmuster in der Zeichenfolge enthalten ist.
„notInLatter()" auf Seite 133	Filtert das erste Array, um festzustellen, ob der Inhalt im zweiten Array enthalten ist. Abhängig vom Wert des Eingabeparameters `in_or_not`, gibt die Funktion den Inhalt des ersten Arrays, einschließlich oder ausschließlich des übereinstimmenden Teils des zweiten Arrays, zurück.
„passOrFail()" auf Seite 134	Vergleicht die erwarteten und die tatsächlichen Werten der vorausgesetzten Eigenschaft und bestimmt, ob die vorausgesetzte Eigenschaft die Prüfung besteht. Die Eingabeparameter können generische Zahlen, Größenangaben in MB oder GB, CPU-Geschwindigkeiten in MHz oder GHz, boolesche Werte oder Zeichenfolgen sein.
„ppread()" auf Seite 135	Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und teilt jede Zeile in der Datei unter Verwendung des als Eingabeparameter angegebenen Trennzeichens, wenn dieses Trennzeichen in der Zeile vorhanden ist.
„readFile()" auf Seite 135	Liest jede Zeile einer Datei in einen Indexeintrag eines Arrays ein.
„unitMGTOG()" auf Seite 136	Addiert den Inhalt eines Arrays, um die Gesamtanzahl an MBs zu errechnen.
„varToString()" auf Seite 136	Erstellt eine Zeichenfolgedarstellung einer Variablen. Die zu prüfende Variable kann eine Zeichenfolge, eine Zahl, ein Scripting-Wörterbuchobjekt, ein Array oder ein Objekt `CheckItem` sein.

allFiles()

Liest die Dateinamen in einem angegebenen Verzeichnis in ein Array ein.

Zweck

Diese Funktion ruft die Liste der Dateien in dem als Eingabeparameter angegebenen Verzeichnis ab und fügt diese dem Array hinzu. Sie gibt das Array zurück.

Syntax

```
allFiles(Dateipfad)
```

Eingabeparameter

String *Dateipfad*
Der Pfad zu dem Verzeichnis, das die Dateien enthält.

Rückgabewerte

Array *Dateiname*
Gibt das Array zurück, das die Dateinamen im angegebenen Verzeichnis enthält.

arrayToString()

Erstellt eine Zeichenfolgedarstellung für das Array.

Zweck

Diese Funktion verwendet das Array, der als Eingabeparameter übergeben wird, und gibt eine Zeichenfolgedarstellung des Inhalts dieses Arrays zurück.

Syntax

```
arrayToString(arr)
```

Eingabeparameter

Array *arr*
Enthält das Array.

Rückgabewerte

String *result*
Gibt eine Zeichenfolgedarstellung des Arrays zurück, wobei die einzelnen Elemente durch Kommas getrennt werden.

bigthan()

Berechnet die Differenz zwischen dem erwarteten und dem tatsächlichen Wert der vorausgesetzten Eigenschaft, wenn diese vorausgesetzte Eigenschaft in MB oder GB angegeben ist.

Zweck

Diese Funktion ruft zuerst die Funktion „changeMG()" auf Seite 126 auf, um die erwarteten und tatsächlichen Werte der vorausgesetzten Eigenschaft in MB zu ändern, sofern dies erforderlich ist. Anschließend prüft sie, ob der zurückgegebene Wert der Funktionen null ist. Wenn der zurückgegebene Wert einer der Funktionen 0MB ist, wird die Funktion beendet. Die Funktion prüft, ob der Wert in MB oder GB angegeben ist, und konvertiert ihn in MB, falls dies erforderlich ist. Sie berech-

net die Differenz zwischen den endgültigen formatierten Werten und gibt das Ergebnis zurück.

Syntax

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

Eingabeparameter

String *expect*
Der erwartete Wert der vorausgesetzten Eigenschaft.

String *real*
Der tatsächliche Wert der vorausgesetzten Eigenschaft.

Rückgabewerte

String *bigthan*
Gibt die Differenz in MB zurück bzw. 0MB, wenn die Werte identisch sind.

changeMG()

Formatiert die Eingabeparameter, um alle zusätzlichen Zeichen für Zifferngruppierung aus dem Parameter zu entfernen, und gibt den formatierten Parameter zurück, sofern der Eingabeparameter nicht MB oder GB enthält. Wenn der Eingabeparameter MB oder GB enthält, wird er in GB bzw. MB konvertiert.

Zweck

Diese Funktion ruft die Funktion „getDecimalSeparator()" auf Seite 132 auf, um das Dezimaltrennzeichen für die aktuelle Ländereinstellung zu bestimmen, und entfernt dann alle zusätzlichen Zeichen für die Zifferngruppierung für diese Ländereinstellung aus dem Eingabeparameter (Zahl). Anschließend ruft sie die Funktion „getFirstMatch()" auf Seite 132 auf, um festzustellen, oder der Wert in MB oder GB angegeben ist, und konvertiert den Wert dann in GB bzw. MB.

Syntax

```
changeMG(tochange)
```

Eingabeparameter

String *tochange*
Enthält das Wertformat und konvertiert dann den Wert gegebenenfalls.

Rückgabewerte

String *changeMG*
Gibt die formatierte Zahl ohne die Zeichen für die Zifferngruppierung zurück bzw. den Wert in MB oder GB.

checkItemToString()

Erstellt eine Zeichenfolgedarstellung für das Objekt `CheckItem`.

Zweck

Diese Funktion verwendet das Objekt `CheckItem`, das als Eingabeparameter übergeben wird, und gibt eine Zeichenfolgedarstellung zurück, die sich aus den Werten

verschiedener Eigenschaften für diese Instanz des Objekts CheckItem zusammensetzt.

Syntax

```
checkItemToString(var)
```

Eingabeparameter

CheckItem *var*
Enthält die Instanz des Objekts CheckItem.

Rückgabewerte

String *result*
Gibt eine Zeichenfolgedarstellung für die Eigenschaften des Objekts CheckItem zurück, wie z. B.:

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

dictionaryToString()

Erstellt eine Zeichenfolgedarstellung für das Scripting-Wörterbuchobjekt.

Zweck

Diese Funktion verwendet das Wörterbuchobjekt, das als Eingabeparameter übergeben wird, und gibt eine Zeichenfolgedarstellung des Inhalts dieses Wörterbuchobjekts zurück.

Syntax

```
0dictionaryToString()
```

Eingabeparameter

Dictionary *dic*
Enthält das Wörterbuchobjekt.

Rückgabewerte

String *result*
Gibt eine Zeichenfolgedarstellung des Wörterbuchobjekts zurück, wobei die einzelnen Schlüssel und Elemente durch ein Gleichheitszeichen getrennt sind.

exeCommand()

Führt den angegebenen Befehl aus und gibt das Ergebnis dieser Befehlsausführung zurück.

Zweck

Diese Funktion führt den als Eingabeparameter angegebenen Befehl aus. Treten Fehler auf, ruft sie die Subroutine logWarning auf, um die Fehler anzuzeigen. Andernfalls gibt sie das Ergebnis der Ausführung des Befehls zurück.

Syntax

```
exeCommand(cmd)
```

Eingabeparameter

String *cmd*
Der Name des auszuführenden Befehls.

Rückgabewerte

String *result*
Gibt eine Zeichenfolge zurück, die das Ergebnis der Ausführung dieses Befehls enthält.

filterCommand()

Führt den angegebenen Befehl aus und gibt die Zeilen aus dem Ergebnis des Befehls zurück, die mit dem angegebenen Muster übereinstimmen.

Zweck

Diese Funktion führt den als Eingabeparameter angegebenen Befehl aus. Sie analysiert das Ergebnis der Ausführung des Befehls, und prüft, ob eine Zeile aus dem Ergebnis mit dem als Eingabeparameter angegebenen Zeilenmuster übereinstimmt. Wenn es eine Übereinstimmung gibt, ruft sie die Funktion „getFirstMatch()" auf Seite 132 auf, um festzustellen, ob es auch eine Übereinstimmung zwischen der als Eingabeparameter angegebenen Informationszeile und dem Ergebnis des Befehls gibt. Wenn es eine Übereinstimmung gibt, verwendet sie die Funktion `Join`, um den Inhalt des Wörterbuchobjekts aus der Funktion `getFirstMatch()` zurückzugeben.

Syntax

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

Eingabeparameter

String *cmd*
Der Name des auszuführenden Befehls.

String *line_patt*
Das Zeilenmuster, nach dem im Ergebnis der Ausführung des Befehls gesucht werden soll.

Number *after_line*
Die Anzahl der Zeilen, nach der die Suche nach dem Informationsmuster gestoppt wird.

String *info_patt*
Das Informationsmuster, nach dem in jeder Zeile des Befehlsergebnisses gesucht werden soll.

Rückgabewerte

String *filterCommand*
Gibt den Inhalt des Wörterbuchobjekts als einzelne Zeichenfolge zurück.

filterFile()

Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und filtert ihn.

Zweck

Diese Funktion liest jede Zeile der Datei ein und übergibt jede Zeile mit dem Suchmuster an die Funktion „getFirstMatch()" auf Seite 132. Wenn eine Übereinstimmung zurückgegeben wird und die Zeile noch nicht im Wörterbuchobjekt vorhanden ist, wird die Zeile in das Wörterbuchobjekt geschrieben. Die Funktion arbeitet in einer Schleife die gesamte Datei ab und gibt dann beim Erreichen des Dateiendes das Wörterbuchobjekt zurück.

Syntax

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

Eingabeparameter

String *fileName*
Die zu filternde Datei.

String *patt*
Das Muster, nach dem in jeder Zeile der Datei gesucht werden soll.

Rückgabewerte

Dictionary *dic.keys*
Gibt das Wörterbuchobjekt *dic* mit den aus der Datei gefilterten Zeilen zurück.

findNewest()

Sucht die neueste Konfigurationsdatei in einem Array.

Zweck

Diese Funktion arbeitet das Array in einer Schleife ab und bestimmt, welche Datei im Array die neueste Konfigurationsdatei ist. Sie gibt den Namen der Datei zurück.

Syntax

```
findNewest(arr)
```

Eingabeparameter

Array *arr*
Enthält die zu prüfende Gruppe von Konfigurationsdateien.

Rückgabewerte

String *result*
Gibt den Namen der neuesten Konfigurationsdatei zurück.

findSuitableFile()

Sucht die relevante Konfigurationsdatei für ein Produkt und eine Version.

Zweck

Diese Funktion ruft die Funktion „getFirstMatch()" auf Seite 132 auf, um die Gruppe von Dateien, die die als Eingabeparameter angegebene Dateierweiterung haben, aus der von der Funktion „allFiles()" auf Seite 124 zurückgegebenen Liste von Dateien abzurufen. Anschließend ruft sie die Funktion „getFirstMatch()" auf Seite 132

erneut auf, um die Gruppe von Dateien zurückzugeben, die den als Eingabeparameter angegebenen Produktcode im Dateinamen enthalten. Sie ruft dieselbe Funktion auf, um die Gruppe von Dateien abzurufen, die die als Eingabeparameter angegebene Version im Dateinamen enthalten. Wenn die Funktion eine oder mehrere Dateiübereinstimmungen für die Version findet, ruft sie die Funktion „findNewest()" auf Seite 129 auf, um die neueste Version dieser Datei abzurufen, und gibt diesen Dateinamen zurück. Andernfalls gibt sie die Datei `common.bat` zurück oder verwendet die Subroutinen `logScreen` und `logWarning`, bevor sie die neuerste Version der Konfigurationsdatei für den Produktcode zurückgibt.

Syntax

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

Eingabeparameter

String *pd*
Der Produktcode, der der zu suchenden Datei zugeordnet ist und der in der Produktcodedatei *IPS-Stammverzeichnis*`/codename.cfg` angegeben ist.

String *version*
Die Version des Produkts, die der zu suchenden Datei zugeordnet ist. <*Version*> ist der achtstellige Code, in dem die Version, das Release, die Modifikation und die Stufe mit jeweils zwei Ziffern dargestellt werden, z. B. Version 7.3.21 ist 07032100.

String *suf*
Die Erweiterung für den Typ der zu suchenden Datei, z. B. `cfg` oder `.bat`.

String *filepath*
Der Pfad zu dem Verzeichnis, das die zu suchende Datei enthält.

Rückgabewerte

String *findSuitableFile*
Gibt je nach Ergebnissen der aufgerufenen Funktionen einen der folgenden Dateinamen zurück:

- *PD-Version*`.cfg`: Die neueste Version der Datei für den zugeordneten Produktcode und die zugeordnete Version.
- `common.bat`: Wenn die als Eingabeparameter angegebene Dateierweiterung `.bat` ist.
- *pd*`.cfg`: Die neueste Version der generischen Konfigurationsdatei für das Produkt, wenn keine Datei gefunden wird, die die angegebene Version enthält.

fmt()

Ändert eine Zeichenfolge, indem der Zeichenfolge eine bestimmte Anzahl an Zeichen aus einer anderen Zeichenfolge hinzugefügt und die andere Zeichenfolge mit Leerzeichen aufgefüllt wird, wenn die andere Zeichenfolge zu kurz ist, bzw. die andere Zeichenfolge gekürzt wird, wenn sie zu lang ist.

Zweck

Diese Funktion sucht nach dem Ausdruck `%#s` im Eingabeparameter *s* des Typs Zeichenfolge. Der Ausdruck `%#s` bestimmt die angegebene Zeichenanzahl # des Eingabeparameters *args*, die der ersten Zeichenfolge an der Position dieses Ausdrucks hinzugefügt werden. Wenn die angegebene Anzahl größer ist als Länge des Eingabeparameters *args*, wird die Differenz mit Leerzeichen aufgefüllt. Wenn die

angegebene Anzahl kleiner ist als die Länge des Eingabeparameters *args*, wird die Länge um die Differenz gekürzt. Wenn die angegebene Anzahl 0 ist, wird die vollständige Länge des Eingabeparameters *args* zur ersten Zeichenfolge an der entsprechenden Position in der Zeichenfolge hinzugefügt.

Syntax

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

Eingabeparameter

String *s*
Enthält die Zeichenfolge, die um die angegebene Zeichenanzahl im Ausdruck %#s in dieser Zeichenfolge geändert werden soll.

Array *args*
Enthält die Gruppe von Zeichen, die den Eingabeparameter *s* ändern.

Rückgabewerte

String *result*
Gibt die geänderte Zeichenfolge zurück.

Beispiel

```
fmt("Hello %5s!",array("Neo")) returns "Hello Neo  !" wird mit zusätzlichen Leerzeichen aufgefüllt.
fmt("Hello %5s!",array("Mr. Anderson")) wird auf "Mr. A" gekürzt und gibt dann "Hello Mr. A!" zurück.
fmt("Hello %0s!",array("Mr. Anderson")) gibt "Hello Mr. Anderson!" zurück.
```

formatForDisplay()

Formatiert den Eingabeparameter, um ihn lesbar machen.

Zweck

Diese Funktion ruft die Funktion „formatSizeForDisplay()" auf, um den Eingabeparameter zu formatieren.

Syntax

```
formatForDisplay(val)
```

Eingabeparameter

Variable *val*
Die zu formatierende Variable.

Rückgabewerte

String *vartoString*
Gibt das Ergebnis der aufgerufenen Funktion „formatSizeForDisplay()" zurück.

formatSizeForDisplay()

Akzeptiert den Eingabeparameter und fügt dem Eingabeparameter zwei Nachkommastellen hinzu bzw. kürzt die Nachkommastellen des Eingabeparameters auf zwei Dezimalstellen. Aus 123MB wird beispielsweise 123,00MB und aus 12,123MBs wird 12,12MBs.

Zweck

Diese Funktion zählt die Anzahl der Zeichen im Eingabeparameter, prüft, ob es sich um eine Zahl oder eine Zeichenfolge handelt, und teilt den Eingabeparameter in ganze Teile und Bruchteile. Abhängig vom Bruchteil fügt sie zwei Dezimalstellen hinzu oder kürzt den Bruchteil auf zwei Dezimalstellen. Sie gibt das Ergebnis zurück.

Syntax

```
formatSizeForDisplay(size)
```

Eingabeparameter

Integer *size*
Der Wert, der auf zwei Dezimalstellen gerundet werden soll.

Rückgabewerte

Integer *val*
Gibt den auf zwei Dezimalstellen gerundeten Wert zurück.

getDecimalSeparator()

Bestimmt das Dezimaltrennzeichen, das für die aktuelle Ländereinstellung verwendet wird.

Zweck

Diese Funktion erstellt eine Bruchzahl und verwendet dann die Funktion `Mid()`, um das Dezimaltrennzeichen, das in dieser Bruchzahl verwendet wird, zu bestimmen.

Syntax

```
getDecimalSeparator()
```

Eingabeparameter

None

Rückgabewerte

Character *sep*
Gibt das Dezimaltrennzeichen, z. B. , (Komma) oder . (Punkt), für die Ländereinstellung zurück.

getFirstMatch()

Ruft die erste Übereinstimmung für den Suchbegriff im Array ab.

Zweck

Diese Funktion verwendet einen regulären Ausdruck für die Suche des Musters, das als Eingabeparameter übergeben wird, in dem Array, das ebenfalls als Eingabeparameter übergeben wird. Wenn sie die erste Übereinstimmung des Musters im Array findet, fügt sie den Wert aus dem Array dem Scripting-Wörterbuchobjekt hinzu.

Syntax

```
getFirstMatch(patt, arr)
```

Eingabeparameter

String *patt*
Enthält das Muster, nach dem gesucht wird.

Array *arr*
Enthält das Array, in dem nach dem Suchmuster gesucht wird.

Rückgabewerte

Dictionary *keys*
Gibt die Schlüssel für das Scripting-Wörterbuchobjekt zurück.

isMatch()

Prüft, ob das Suchmuster in der Zeichenfolge enthalten ist.

Zweck

Diese Funktion ruft die Funktion „getFirstMatch()" auf Seite 132 auf und übergibt dabei das Muster und die Zeichenfolge (in einem Array) als Eingabeparameter an diese Funktion. Sie ruft die Funktion `ubound` auf, um zu prüfen, ob der zurückgegebene Wert der Funktion `getFirstMatch()` größer-gleich 0 ist. Ist dies der Fall, handelt es sich um eine Übereinstimmung, andernfalls nicht.

Syntax

```
isMatch(patt,str)
```

Eingabeparameter

String *patt*
Enthält das Muster, nach dem gesucht wird.

String *str*
Enthält die Zeichenfolge, in der nach dem Suchmuster gesucht wird.

Rückgabewerte

Boolean *True|False*
Gibt `True` zurück, wenn es eine Übereinstimmung gibt, andernfalls `False`.

notInLatter()

Filtert das erste Array, um festzustellen, ob der Inhalt im zweiten Array enthalten ist. Abhängig vom Wert des Eingabeparameters `in_or_not`, gibt die Funktion den Inhalt des ersten Arrays, einschließlich oder ausschließlich des übereinstimmenden Teils des zweiten Arrays, zurück.

Zweck

Syntax

```
notInLatter(arr1, arr2, in_or_not)
```

Eingabeparameter

Array *arr1*
Quelle der Kopie

Array *arr2*
Ziel der Kopie

String *in_or_out*
Enthält entweder "in" oder "not", je nachdem, ob die Funktion den Inhalt des ersten Arrays so gefiltert zurückgeben soll, dass nur die Inhalte, die dem zweiten Array ("in") entsprechen, oder die Inhalte, die nicht mit dem zweiten Array ("not") übereinstimmen, in der Rückgabe enthalten sind.

Rückgabewerte

Dictionary *keys*
Gibt die Schlüssel des Scripting-Wörterbuchobjekts zurück, das das erste Array so gefiltert enthält, dass es nur den Inhalt enthält, der dem zweiten Array (in_or_not = "in") entspricht, oder den Inhalt, der nicht mit dem zweiten Array (in_or_not = "not") übereinstimmt.

---

passOrFail()

Vergleicht die erwarteten und die tatsächlichen Werten der vorausgesetzten Eigenschaft und bestimmt, ob die vorausgesetzte Eigenschaft die Prüfung besteht. Die Eingabeparameter können generische Zahlen, Größenangaben in MB oder GB, CPU-Geschwindigkeiten in MHz oder GHz, boolesche Werte oder Zeichenfolgen sein.

Zweck

Diese Funktion ruft zuerst die Funktion „changeMG()" auf Seite 126 auf, um die erwarteten und tatsächlichen Werte zu formatieren und ggf. zu konvertieren. Sie prüft, ob ein Wert 0 ist, und gibt, wenn ja, "FAIL" zurück und endet. Wenn die Werte ungleich 0 sind, prüft die Funktion, ob die Werte boolesche Werte, numerische Werte, Größenwerte in MB oder GB, Werte für CPU-Geschwindigkeit in MHz (nur Windows) oder GHz oder Zeichenfolgen sind. Anschließend vergleicht sie die Werte und gibt das Ergebnis zurück.

Syntax

```
passOrFail(expect,real)
```

Eingabeparameter

String *expect*
Der erwartete Wert für die vorausgesetzte Eigenschaft.

String *real*
Der tatsächliche Wert für die vorausgesetzte Eigenschaft.

Rückgabewerte

String *passOrFail*
Gibt "PASS" oder "FAIL" zurück, abhängig davon, ob der erwartete Wert größer-gleich dem tatsächlichen Wert ist.

ppread()

Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und teilt jede Zeile in der Datei unter Verwendung des als Eingabeparameter angegebenen Trennzeichens, wenn dieses Trennzeichen in der Zeile vorhanden ist.

Zweck

Diese Funktion liest jede Zeile der Datei, entfernt alle führenden und nachfolgenden Leerzeichen und überprüft, ob die Zeile das Trennzeichen enthält. Wenn die Zeile das Trennzeichen enthält, wird sie durch das Trennzeichen aufgeteilt, indem jeder Teil als Element im Wörterbuchobjekt hinzugefügt wird. Enthält die Zeile das Trennzeichen nicht, wird die getrimmte Zeile dem Wörterbuchobjekt hinzugefügt. Die Funktion gibt ein Array zurück, das das Wörterbuchobjekt als ersten Index enthält.

Syntax

```
ppread(fileName, sep)
```

Eingabeparameter

String *fileName*
Der Name der Datei, die in das Wörterbuchobjekt eingelesen werden soll.

Character *sep*
Das Zeichen, das das Trennzeichen darstellt, mit dem eine Zeile in der Datei aufgeteilt wird.

Rückgabewerte

Array *array(dic)*
Gibt ein Array mit dem Wörterbuchobjekt (`dic`) als ersten Index zurück.

Beispiel

Nicht verfügbar.

readFile()

Liest jede Zeile einer Datei in einen Indexeintrag eines Arrays ein.

Zweck

Diese Funktion öffnet die Datei und liest jede Zeile der Datei in einen Indexeintrag des Arrays ein. Sie gibt das Array zurück.

Syntax

```
readFile(fileName)
```

Eingabeparameter

String *fileName*
Der Name der in das Array einzulesenden Datei.

Rückgabewerte

Array *fileContents*
Gibt das Array mit dem Inhalt der Datei zurück.

unitMGTOG()

Addiert den Inhalt eines Arrays, um die Gesamtanzahl an MB zu errechnen.

Zweck

Diese Funktion konvertiert den Wert jedes Index im Array in MB und addiert dann die Werte.

Syntax

```
unitMGTOG(arr)
```

Eingabeparameter

Array *arr*
Enthält das Array.

Rückgabewerte

String *unitMGTOG*
Gibt die Summe des Inhalts des Arrays in MB zurück und fügt "MB" an die Summe an.

varToString()

Erstellt eine Zeichenfolgedarstellung einer Variablen. Die zu prüfende Variable kann eine Zeichenfolge, eine Zahl, ein Scripting-Wörterbuchobjekt, ein Array oder ein Objekt CheckItem sein.

Zweck

Diese Funktion prüft die Daten oder den Objekttyp der Variablen und ruft die entsprechende Funktion auf, um eine Zeichenfolgedarstellung für diese Daten bzw. diesen Objekttyp zu erstellen.

Tabelle 32. Für jeden Variablentyp aufgerufene Funktion

Variablentyp	Aufgerufene Funktion
Array	„arrayToString()" auf Seite 125
Objekt CheckItem	„checkItemToString()" auf Seite 126
Scripting-Wörterbuchobjekt	„dictionaryToString()" auf Seite 127

Syntax

```
varToString(var)
```

Eingabeparameter

Variable *var*
Die unterstützten Variablen sind Zeichenfolge, Zahl, Scripting-Wörterbuchobjekt, Array oder CheckItem-Objekt.

Rückgabewerte

String *vartoString*
Gibt eine Zeichenfolgedarstellung der Variablen, einschließlich der von allen aufgerufenen Funktionen zurückgegebenen Werte, zurück.

Anhang F. Subroutinen des Protokolldienstprogramms auf Windows-Systemen

IBM Prerequisite Scanner enthält eine Gruppe allgemeiner Protokollierungssubroutinen in der Datei `preq.vbs`, mit denen Nachrichten am Bildschirm angezeigt oder in eine Protokolldatei geschrieben werden können.

In Tabelle 33 sind die Protokolldienstprogramme beschrieben.

Tabelle 33. Subroutinen des Protokolldienstprogramms

Subroutine	Beschreibung	Eingabeparameter
`deleteLogFile`	Löscht die Protokolldatei, wenn sie vorhanden ist.	Ohne
`log(level, msg)`	Schreibt die Nachricht mit der Funktion „fmt()" auf Seite 130 in die Protokolldatei. Das Protokoll enthält außerdem das aktuelle Datum und die aktuelle Uhrzeit.	• `level`: Eine Zeichenfolge, die den Typ der Nachricht festlegt, z. B. Information oder Warnung. • `msg`: Eine Zeichenfolge, die die zu protokollierende Nachricht darstellt.
`logDebug(msg)`	Ruft die Funktion `log()` auf und übergibt dabei `"DEBUG"` als Eingabeparameter für die Protokollierungsstufe (level).	`msg`: Eine Zeichenfolge, die die zu protokollierende Nachricht darstellt.
`logError(msg)`	Ruft die Funktion `log()` auf und übergibt dabei `"ERROR"` als Eingabeparameter für die Protokollierungsstufe (level).	`msg`: Eine Zeichenfolge, die die zu protokollierende Nachricht darstellt.
`logInfo(msg)`	Ruft die Funktion `log()` auf und übergibt dabei `"INFO"` als Eingabeparameter für die Protokollierungsstufe (level).	`msg`: Eine Zeichenfolge, die die zu protokollierende Nachricht darstellt.
`logScreen(msg)`	Gibt die Nachricht am Bildschirm aus.	`msg`: Eine Zeichenfolge, die die am Bildschirm auszugebende Nachricht darstellt.
`logScreenWith Replacement (msg, replaceStr)`	Gibt die Nachricht am Bildschirm aus und übergibt dabei einen Nachrichtencode und eine Zeichenfolge als Eingabeparameter.	• `msg`: Nachrichtencode, der die am Bildschirm auszugebende Nachrichtenzeichenfolge darstellt. • `replaceStr`: Die Zeichenfolge, die *%variable* im Nachrichtencodewert ersetzt.
`logScreenWith MultiReplacements (msg, replaceStrArray)`	Gibt die Nachricht am Bildschirm aus und übergibt dabei einen Nachrichtencode und ein Zeichenfolgenarray als Eingabeparameter.	• `msg`: Nachrichtencode, der die am Bildschirm auszugebende Nachrichtenzeichenfolge darstellt. • `replaceStrArray`: Das Zeichenfolgenarray, in dem jeder Index ein Element *%variable* im Nachrichtencodewert ersetzt.
`logWarning(msg)`	Ruft die Funktion `log()` auf und übergibt dabei `"WARNING"` als Eingabeparameter für die Protokollierungsstufe (level).	`msg`: Eine Zeichenfolge, die die zu protokollierende Nachricht darstellt.

Anhang G. Subroutinen des Dateidienstprogramms auf Windows-Systemen

Prerequisite Scanner enthält eine Gruppe allgemeiner Dateisubroutinen in der Datei `/lib/common_function.vbs`, mit denen Dateien bearbeitet werden können. Darüber hinaus enthält das Produkt eine Reihe von Funktionen für die Bearbeitung von Dateien.

In Tabelle 34 sind die Dienstprogramme für Dateien beschrieben.

Tabelle 34. Subroutinen des Dateidienstprogramms

Subroutine	Beschreibung	Eingabeparameter
`appendToFile(text, fileName)`	Fügt den Text an das Ende der angegebenen Datei an.	• `text`: Eine Zeichenfolge, die den an die Datei anzufügenden Text enthält. • `filename`: Eine Zeichenfolge, die den Namen der zu ändernden Datei darstellt.
`writeToFile(text, fileName)`	Schreibt den Text in die angegebene Datei und überschreibt ggf. vorhandenen Inhalt.	• `text`: Eine Zeichenfolge, die den in die Datei zu schreibenden Text enthält. • `filename`: Eine Zeichenfolge, die den Namen der zu ändernden Datei darstellt.

In Tabelle 35 sind die Dateifunktionen beschrieben, die Dateien bearbeiten.

Tabelle 35. Dienstprogrammfunktionen für Dateien

Funktion	Beschreibung
„allFiles()" auf Seite 124	Liest die Dateinamen in einem angegebenen Verzeichnis in ein Array ein.
„filterFile()" auf Seite 128	Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und filtert ihn.
„findNewest()" auf Seite 129	Sucht die neueste Konfigurationsdatei.
„findSuitableFile()" auf Seite 129	Sucht die relevante Konfigurationsdatei für ein Produkt und eine Version.
„ppread()" auf Seite 135	Liest den Inhalt einer Datei in ein Scripting-Wörterbuchobjekt ein und teilt jede Zeile in der Datei unter Verwendung des als Eingabeparameter angegebenen Trennzeichens, wenn dieses Trennzeichen in der Zeile vorhanden ist.
„readFile()" auf Seite 135	Liest jede Zeile einer Datei in einen Indexeintrag eines Arrays ein.

Anhang H. Weitere allgemeine Funktionen und Subroutinen für Windows-Systeme

Prerequisite Scanner enthält eine Gruppe weiterer allgemeiner Funktionen und Subroutinen, die in verschiedenen Dateien verwendet werden.

Beschreibung der weiteren allgemeinen Funktionen und Subroutinen

Tabelle 36. Weitere allgemeine Funktionen und Subroutinen für Windows-Systeme

Funktion oder Subroutine	Beschreibung
„ffirstMatch()"	Ruft die erste Übereinstimmung für den Suchbegriff im Array ab.
„getValue()" auf Seite 142	Ruft den verfügbaren Plattenspeicherplatz für ein bestimmtes Verzeichnis ab.
„removeSpecialCharacters()" auf Seite 143	Entfernt die Marken- und anderen Sonderzeichen, um Vergleiche zu vereinfachen.
„versionCompare()" auf Seite 143	Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob die erforderliche Eigenschaft die Prüfung der Voraussetzungen besteht. Die Funktion erwartet durch Punkte getrennte Versionszeichenfolgen als Eingabeparameter, z. B. 1.0.0.4, 2.3, 3.40.26.7800 oder 2.3.*.

ffirstMatch()

Ruft die erste Übereinstimmung für den Suchbegriff im Array ab.

Zweck

Diese Funktion verwendet einen regulären Ausdruck für die Suche des Musters, das als Eingabeparameter übergeben wird, in dem Array, das ebenfalls als Eingabeparameter übergeben wird. Wenn sie die erste Übereinstimmung des Musters im Array findet, fügt sie den Wert aus dem Array dem Scripting-Wörterbuchobjekt hinzu.

Übergeordnete Funktionen

Tabelle 37. Übergeordnete Funktionen, die ffirstMatch() aufrufen

Übergeordnete Funktion, Script	Beschreibung
`ud620db2level(expect, real)` in `DB2_Version_compare.vbs`	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die DB2-Version.
`oslevelcompare(expect, real)` in `OS_Version_compare.vbs`	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die Version des Betriebssystems.

Syntax

```
ffirstmatch(patt,arr)
```

Eingabeparameter

String *patt*
Enthält das Muster, nach dem gesucht wird.

Array *arr*
Enthält das Array, in dem nach dem Suchmuster gesucht wird.

Rückgabewerte

Dictionary *keys*
Gibt die Schlüssel für das Scripting-Wörterbuchobjekt zurück.

getValue()

Ruft den verfügbaren Plattenspeicherplatz für ein bestimmtes Verzeichnis ab.

Zweck

Diese Subroutine verwendet die Instanz des Dateisystemobjekts, um die Funktion `getDriveName()` für den Pfadeingabeparameter aufrufen, und verwendet dann die Eigenschaft `freeSpace`, um den verfügbaren Plattenspeicherplatz abzurufen, der dann in MB konvertiert wird. Der Eingabeparameter für die vorausgesetzte Eigenschaft und dessen Wert werden in die temporäre Textdatei geschrieben, die der Scriptdatei zugeordnet ist.

Scripts

Tabelle 38. Scripts, die getValue() verwenden

Script	Beschreibung
`DEZ_01040000.vbs`	Script, das die erforderlichen Eigenschaften erfasst und nur in der Konfigurationsdatei DEZ 01040000 bereitstellt
`LCM_TAD_common.vbs`	Script, das die erforderlichen Eigenschaften erfasst und nur in den Konfigurationsdateien LCM 02300000 und TAD 07200000 bereitstellt
`TAD722_impl.vbs`	Script, das die erforderlichen Eigenschaften erfasst und nur in der Konfigurationsdatei TAD 07220000 bereitstellt

Syntax

```
getValue fso, sKey, drvPath
```

Eingabeparameter

File system object *fso*
Instanz des Dateisystemobjekts

String *sKey*
Enthält eine Zeichenfolge mit dem Namen der erforderlichen Eigenschaft und dem Gleichheitszeichen

String *drvPath*
Enthält den Pfad, für den der verfügbare Plattenspeicherplatz abgerufen werden soll

Rückgabewerte

None

removeSpecialCharacters()

Entfernt die Marken- und anderen Sonderzeichen, um Vergleiche zu vereinfachen. Die Funktion ist in der Datei `/lib/common.vbs` enthalten.

Zweck

Diese Funktion ruft die Funktion `Ersetzen()` auf, um Marken-, Copyright- und registrierte Symbole durch `""` zu ersetzen.

Syntax

```
removeSpecialCharacters(s)
```

Eingabeparameter

String *s*
Enthält die Zeichenfolge, aus der die Zeichen entfernt werden müssen.

Rückgabewerte

String *s*
Gibt die Zeichenfolge ohne Sonderzeichen zurück.

versionCompare()

Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob die erforderliche Eigenschaft die Prüfung der Voraussetzungen besteht. Die Funktion erwartet durch Punkte getrennte Versionszeichenfolgen als Eingabeparameter, z. B. 1.0.0.4, 2.3, 3.40.26.7800 oder 2.3.*.

Zweck

Diese Funktion behandelt zuerst die Sonderfälle, in denen einer oder beide Eingabeparameter leer sind, und gibt Rückgabecodes zurück, um diese Fälle darzustellen. Sie teilt jede Version durch Punkte in mehrere Teile ein. Wenn der letzte Teil der Version das Platzhalterzeichen * ist, interpretiert die Funktion alle fehlenden Teile der Version als das Platzhalterzeichen. 2.* entspricht beispielsweise 2.1 und 2.3.*. Sie führt dann eine Schleife durch die Liste der Teile für jede Version aus und vergleicht sie. Dann gibt sie Rückgabecodes zurück, die abhängig davon sind, ob der erwartete Wert kleiner als, gleich oder größer als der tatsächliche Wert ist.

Übergeordnete Funktionen

Tabelle 39. Übergeordnete Funktionen, die versionCompare aufrufen

Übergeordnete Funktion, Script	Beschreibung
`cygwinVersion_compare.vbs`	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die cygwin-Version.

Tabelle 39. Übergeordnete Funktionen, die versionCompare aufrufen (Forts.)

Übergeordnete Funktion, Script	Beschreibung
gskit7Version_compare.vbs	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die gskit Version 7.
gkit8Version_compare.vbs	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die gskit Version 8.
internetExplorer.version_compare.vbs	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die Version von Internet Explorer.
os.servicePack_compare.vbs	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für das Service-Pack des Betriebssystems.
os.versionNumber_compare.vbs	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die Version des Betriebssystems.

Syntax

```
versionCompare(ver1,ver2)
```

Eingabeparameter

String *ver1*
Enthält die erwartete Version für eine erforderliche Eigenschaft.

String *ver2*
Enthält die tatsächliche Version für eine erforderliche Eigenschaft.

Rückgabewerte

Integer *0*
Gibt den Rückgabecode 0 zurück, wenn beide Eingabeparameter gleich sind. Die übergeordnete Funktion gibt "PASS" zurück.

Sonderfall: Gibt den Rückgabecode 0 zurück und endet, wenn beide Eingabeparameter leer sind.

Integer *-1*
Gibt den Rückgabecode -1 zurück, wenn der erste Eingabeparameter kleiner ist als der zweite Eingabeparameter. Die übergeordnete Funktion gibt "FAIL" zurück.

Sonderfall: Gibt den Rückgabecode "-1" zurück und endet, wenn der erste Eingabeparameter leer ist.

Integer *1*
Gibt den Rückgabecode -1 zurück, wenn der erste Eingabeparameter größer ist als der zweite Eingabeparameter. Die übergeordnete Funktion gibt "PASS" zurück.

Sonderfall: Gibt den Rückgabecode "1" zurück und endet, wenn der zweite Eingabeparameter leer ist.

Anhang I. Allgemeine Funktionen für UNIX-Systeme

Prerequisite Scanner enthält eine Gruppe allgemeiner Funktionen in der Datei `/lib/common_function.sh` für die Durchführung von Prüfungen auf UNIX-basierten Systemen.

Tabelle 40. Funktionen in common_function.sh

Funktion	Beschreibung
„AddMG()" auf Seite 146	Prüft, ob die Eingabeparameter in MB oder GB angegeben sind, und fügt die Parameter hinzu.
„changeMG()"	Konvertiert den Eingabeparameter für die erforderlichen Eigenschaften für den Plattenspeicherplatz und den Hauptspeicher in MB oder GB.
„compare()" auf Seite 147	Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob der erste Wert (tatsächlich) kleiner als der zweite Wert (erwartet) ist.
„cutdown()" auf Seite 147	Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob der erste Wert (tatsächlich) kleiner als der zweite Wert (erwartet) ist. Anschließend wird die Differenz zwischen den beiden Werten ausgegeben, wenn der erste Wert nicht kleiner als der zweite Wert ist.
„findOSInfo()" auf Seite 149	Ermittelt Betriebssystemversion, Release-Level und Release-Level-Version des Betriebssystems und die Hardwareimplementierungsdaten für das System.
„mes4path()" auf Seite 148	Ermittelt den freien Plattenspeicherplatz für jedes angehängte Dateisystem.
„mes4Path1()" auf Seite 149	Ermittelt den freien Plattenspeicherplatz für jedes angehängte Dateisystem (nur Solaris-Systeme).
„NFScheck()" auf Seite 150	Überprüft den NFS-Status von Mounts auf einem UNIX-basierten System.
„telnetNFS()" auf Seite 150	Prüft, ob die IP-Adresse eines angehängten Dateisystems mit Telnet über den Standardport 2049 erreichbar ist.

changeMG()

Konvertiert den Eingabeparameter für die erforderlichen Eigenschaften für den Plattenspeicherplatz und den Hauptspeicher in MB oder GB.

Zweck

Diese Funktion prüft zunächst, ob die Funktion einen Eingabeparameter empfängt. Wenn sie einen Eingabeparameter empfängt, bestimmt sie, ob der Wert in MB oder GB angegeben ist, und konvertiert den Wert dann in GB bzw. MB.

Syntax

```
changeMG val
```

Eingabeparameter

String *$val*
Enthält den Wert für Plattenspeicherplatz oder Hauptspeicher in MB oder GB.

Rückgabewerte

Integer *1*
Gibt 1 zurück, wenn die Funktion keinen Eingabeparameter empfängt.

String *printf "%.0fM%s",mm[1]*1024,mm[2];*
Gibt den Wert in MB zurück.

String *printf "%.2fG%s",mm[1],mm[2];*
Gibt den Wert in GB zurück.

AddMG()

Prüft, ob die Eingabeparameter in MB oder GB angegeben sind, und fügt die Parameter hinzu.

Zweck

Diese Funktion prüft zunächst, ob die Funktion Eingabeparameter empfängt. Wenn sie Eingabeparameter empfängt, prüft sie, ob der Wert in MB oder GB angegeben ist, und fügt dann die Werte hinzu.

Syntax

```
AddMG val1 val2
```

Eingabeparameter

String *$val1*
Enthält den Wert für Plattenspeicherplatz oder Hauptspeicher in MB oder GB, der dem anderen Eingabeparameter hinzugefügt werden soll.

String *$val2*
Enthält den Wert für Plattenspeicherplatz oder Hauptspeicher in MB oder GB, der dem anderen Eingabeparameter hinzugefügt werden soll.

Rückgabewerte

Integer *1*
Gibt 1 zurück, wenn die Funktion nicht zwei Eingabeparameter empfängt.

String *val*
Gibt die hinzugefügten Werte in MB oder GB zurück.

compare()

Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob der erste Wert (tatsächlich) kleiner als der zweite Wert (erwartet) ist.

Zweck

Diese Funktion prüft zunächst, ob die Funktion zwei Eingabeparameter empfängt. Wenn sie zwei Eingabeparameter empfängt und beide nicht falsch sind, stellt sie fest, ob die Werte in MB oder GB angegeben sind, und vergleicht die beiden Werte dann, um festzustellen, ob der erste Wert kleiner als der zweite Wert ist. Wenn ja, gibt die Funktion einen Wert für falsch zurück. Wenn nicht, gibt sie einen Wert für erfolgreiche Ausführung zurück.

Syntax

```
compare real expected
```

Eingabeparameter

String *$real*
Enthält den tatsächlichen Wert für eine erforderliche Eigenschaft.

String *$expected*
Enthält den erwarteten Wert für eine erforderliche Eigenschaft.

Rückgabewerte

Integer *1*
Gibt 1 zurück, wenn die Funktion nicht zwei Eingabeparameter empfängt.

String *"FAIL|PASS"*
Gibt die Zeichenfolge "FAIL" zurück, wenn der tatsächliche Wert kleiner als der erwartete Wert ist, andernfalls sie gibt die Zeichenfolge "PASS" zurück.

cutdown()

Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht sie, um festzustellen, ob der erste Wert (tatsächlich) kleiner als der zweite Wert (erwartet) ist. Anschließend wird die Differenz zwischen den beiden Werten ausgegeben, wenn der erste Wert nicht kleiner als der zweite Wert ist.

Zweck

Diese Funktion prüft zunächst, ob die Funktion zwei Eingabeparameter empfängt. Wenn die Funktion zwei Eingabeparameter empfängt, bestimmt sie, ob die Werte in MB oder GB angegeben sind, und konvertiert sie dann in MB, wenn sie in GB angegeben sind. Anschließend vergleicht sie die beiden Werte, um zu prüfen, ob der erste Wert kleiner als der zweite Wert ist. Wenn ja,, gibt die Funktion den Wert "0MB" zurück. Andernfalls gibt sie die Differenz zwischen den beiden Werten in MB zurück.

Syntax

```
cutdown real expected
```

Eingabeparameter

String *$real*
Enthält den tatsächlichen Wert für eine erforderliche Eigenschaft.

String *$expected*
Enthält den erwarteten Wert für eine erforderliche Eigenschaft.

Rückgabewerte

Integer *1*
Gibt 1 zurück, wenn die Funktion nicht zwei Eingabeparameter empfängt.

String *"FAIL|PASS"*
Gibt die Zeichenfolge "FAIL" zurück, wenn der tatsächliche Wert kleiner als der erwartete Wert und keiner der Werte in MB oder GB angegeben ist. Andernfalls gibt sie die Zeichenfolge "PASS" zurück.

String *"OMB|Real-ExpectedMB"*
Gibt die Zeichenfolge "OMB" zurück, wenn der tatsächliche Wert kleiner als der erwartete Wert ist, andernfalls gibt sie eine Zeichenfolgedarstellung der Differenz zwischen den beiden konvertierten Werte in MB zurück.

mes4path()

Ermittelt den freien Plattenspeicherplatz für jedes angehängte Dateisystem.

Zweck

Diese Funktion verwendet einen Pfad als Eingabe, ruft den Befehl **uname** auf, um das Betriebssystem zu ermitteln, und ruft dann die Funktion NFScheck auf, um festzustellen, ob das System und die Mounts betriebsbereit sind. Anschließend ruft sie den Befehl df auf, um den freien Plattenspeicherplatz für jeden Mount in einem System zu bestimmen. Sie gibt den Wert für den freien Plattenspeicherplatz zurück.

Syntax

```
mes4Path path
```

Eingabeparameter

String *$path*
Pfad zum System, dessen freier Plattenspeicherplatz geprüft werden soll.

Rückgabewerte

Integer *1*
Gibt den Rückgabecode 1 zurück, wenn die Funktion keinen Eingabeparameter empfängt.

Integer *2*
Gibt den Rückgabecode 2 zurück, wenn der Eingabeparameter kein Pfad ist.

String *$NF*
Gibt den freien Plattenspeicherplatz für jeden Mount zurück.

String *"$path Server NotAvailable Responding for $path"*
Gibt eine Nachricht zurück, die anzeigt, dass der Server für den Pfad nicht verfügbar ist.

mes4Path1()

Ermittelt den freien Plattenspeicherplatz für jedes angehängte Dateisystem (nur Solaris-Systeme).

Zweck

Diese Funktion verwendet einen Pfad als Eingabe und ruft den Befehl **uname** auf, um festzustellen, ob das Betriebssystem Solaris ist. Anschließend ruft sie den Befehl "df" auf, um den freien Plattenspeicherplatz für jeden Mount auf dem System zu bestimmen. Sie gibt den Wert für den freien Plattenspeicherplatz zurück.

Syntax

```
mes4Path1 path
```

Eingabeparameter

String *$path*
Pfad zum System, dessen freier Plattenspeicherplatz geprüft werden soll.

Rückgabewerte

Integer *1*
Gibt den Rückgabecode 1 zurück, wenn die Funktion keinen Eingabeparameter empfängt.

Integer *2*
Gibt den Rückgabecode 2 zurück, wenn der Eingabeparameter kein Pfad ist.

String *$NF*
Gibt den freien Plattenspeicherplatz für jeden Mount zurück.

findOSInfo()

Ermittelt Betriebssystemversion, Release-Level und Release-Level-Version des Betriebssystems und die Hardwareimplementierungsdaten für das System.

Zweck

Diese Funktion führt den Befehl `uname` aus und analysiert die Ausgabe, um die Betriebssystemversion, das Release-Level und die Release-Version des Betriebssystems und die Hardwareimplementierungsdaten für das System zu ermitteln.

Syntax

```
findOSInfo
```

Eingabeparameter

Ohne

Rückgabewerte

String *$oo*
Die Ausgabe von **uname** ohne die Basissysteminformationen.

String *$kk*
Betriebssystemversion

String *$hh*
Hardwareimplementierung, dargestellt als I für i386-Hardware bzw. Z für s390-Hardware.

String *$rr*
Release-Level des Betriebssystems

String *$vv*
Release-Level-Version des Betriebssystems

telnetNFS()

Prüft, ob die IP-Adresse eines angehängten Dateisystems mit Telnet über den Standardport 2049 erreichbar ist.

Zweck

Diese Funktion verwendet eine IP-Adresse als Eingabe und ruft den Befehl **telnet** auf, um zu testen, ob die Fernverbindung am Standard-Telnet-Port 2049 erfolgreich ist. Der Aufbau der Fernverbindung wird 10 Mal wiederholt. Wenn der Befehl **telnet** scheitert, gibt die Funktion den Wert `"FALSE"` zurück, andernfalls gibt sie den Wert `"PASS"` zurück.

Syntax

```
telnetNFS ipaddr
```

Eingabeparameter

String *$ipaddr*
Die IP-Adresse, mit der geprüft wird, ob eine Telnet-Sitzung hergestellt werden kann.

Rückgabewerte

String *"FALSE|TRUE"*
Gibt das Ergebnis der Telnet-Prüfung zurück. Sie gibt `"TRUE"` zurück, wenn die Prüfung erfolgreich ist, andernfalls gibt sie `"FALSE"` zurück.

NFScheck()

Überprüft den NFS-Status von Mounts auf einem UNIX-basierten System.

Zweck

Diese Funktion verwendet einen Pfad als Eingabe und ruft den Befehl "mount" auf, um die Liste der angehängten Dateisysteme abzurufen. Sie ruft den Befehl **uname** auf, um das Betriebssystem zu bestimmen. Anschließend ruft sie den Befehl **ping** auf, um alle angehängten Systeme mit Ping zu überprüfen. Wenn die Systeme mit Ping erreichbar sind, ruft sie die Funktion **telnetNFS** auf, um zu überprüfen, ob eine Fernverbindung hergestellt werden kann. Wenn die Ping- oder Telnet-Aktion scheitert, gibt die Funktion den Wert `"FALSE"` zurück, andernfalls gibt sie den Wert `"PASS"` zurück.

Syntax

```
NFScheck path
```

Eingabeparameter

String *$path*
Verwendet einen gültigen Pfad zu einem Verzeichnis als Eingabe.

Rückgabewerte

Boolescher Wert *TRUE* **oder** *FALSE*
Gibt TRUE zurück, wenn die NFS-Prüfung erfolgreich ist, d. h., wenn die Funktion die zugeordnete IP-Adresse mit Ping erreichen oder Telnet verwenden kann, um eine Verbindung zur zugeordneten IP-Adresse für jedes Dateisystem herzustellen. Andernfalls gibt die Funktion FALSE zurück.

Beispiel

Dieses Syntaxbeispiel ist der Funktion **mes4Path()** entnommen:

```
# check if it's a path
 path=`echo "$1" | sed -n '/^\//p'`
 if [ -z "$path" ];then
  return 2;
 else
          nfs_check_status=`NFScheck $path`
          if [ "$nfs_check_status" = "TRUE" ]; then
         case `uname` in
     ...
```

Anhang J. Weitere Funktionen für UNIX-Systeme

Prerequisite Scanner enthält eine Gruppe allgemeiner Funktionen in verschiedenen Dateien.

Tabelle 41 enthält eine Beschreibung der Funktionen in mehreren Dateien.

Tabelle 41. Allgemeine Funktionen in mehreren Dateien

Funktion	Beschreibung
„formatSizeDisplay()" auf Seite 154	Akzeptiert den Eingabeparameter und fügt dem Eingabeparameter zwei Nachkommastellen hinzu bzw. kürzt die Nachkommastellen des Eingabeparameters auf zwei Dezimalstellen. Aus 123MB wird beispielsweise 123,00MB und aus 12,123MBs wird 12,12MBs.
„versionCompare()" auf Seite 154	Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht jeden Teil der Version, um festzustellen, ob die erforderliche Eigenschaft die Prüfung der Voraussetzungen besteht.

Tabelle 42 enthält eine Beschreibung der Funktionen in der Datei `UNIX-Linux/TAD722_impl.sh` für die Durchführung von Prüfungen für Tivoli License Compliance Manager und Tivoli Asset Discovery for Distributed.

Tabelle 42. Allgemeine Funktionen in TAD722_impl.sh

Funktion	Beschreibung
„checkSunOS()" auf Seite 157	Prüft, ob die Version des Betriebssystems Solaris für SPARC- oder X86-Plattformen bestimmt ist.
„checkHpux()" auf Seite 156	Prüft, ob die Version des Betriebssystems HP-UX für IA64- oder PARISC-Plattformen bestimmt ist.
„checkLinux()" auf Seite 156	Prüft, ob die Version des Linux-Betriebssystems für System-p-, System-z- oder x86-Plattformen bestimmt ist.
„getSystemId()" auf Seite 158	Ruft verschiedene Betriebssystemfunktionen auf, um das relevante Betriebssystem auf den Plattformen zu suchen.
„getValue()" auf Seite 157	Ruft den Wert für einen Schlüssel in einer angegebenen Datei ab, wenn der Schlüssel vorhanden ist.
„setValue()" auf Seite 157	Setzt den Wert für einen Schlüssel in einer angegebenen Datei, wenn die erforderliche Eigenschaft vorhanden ist.
„copyValue()" auf Seite 158	Ruft den Wert für die erforderliche Eigenschaft (Schlüssel) basierend auf dem Produkt und dem Betriebssystem ab und legt diese fest.

Tabelle 42. Allgemeine Funktionen in TAD722_impl.sh (Forts.)

Funktion	Beschreibung
„parseDirParameter()" auf Seite 159	Analysiert den Parameter aus der Parameterliste für das Flag "-p" des Scanners und speichert den Wert in der Liste.
„getClosestExistingParentDir()" auf Seite 159	Ruft das unmittelbar übergeordnete Verzeichnis oder sich selbst ab.
„printDirSize()" auf Seite 159	Überprüft den NFS-Status des angehängten Dateisystems und ruft dann den Plattenspeicherplatz des Dateisystems bzw. dessen übergeordneten Verzeichnisses ab.

formatSizeDisplay()

Akzeptiert den Eingabeparameter und fügt dem Eingabeparameter zwei Nachkommastellen hinzu bzw. kürzt die Nachkommastellen des Eingabeparameters auf zwei Dezimalstellen. Aus 123 MB wird beispielsweise 123,00 MB und aus 12,123 MB wird 12,12 MB.

Zweck

Diese Funktion zählt die Anzahl der Zeichen im Eingabeparameter, prüft, ob es sich um eine Zahl oder eine Zeichenfolge handelt, und teilt den Eingabeteil in ganze Teile und Bruchteile. Abhängig vom Bruchteil fügt sie zwei Dezimalstellen hinzu oder kürzt den Bruchteil auf zwei Dezimalstellen. Sie gibt das Ergebnis zurück.

Übergeordnete Scripts

Die folgenden Scripts enthalten die Funktion:

- `./Unix-Linux/common.sh`
- `LCM_TAD_common.sh`

Syntax

```
formatSizeDisplay val
```

Eingabeparameter

Integer *$val*
Der Wert, der auf zwei Dezimalstellen gerundet werden soll.

Rückgabewerte

Integer *val*
Gibt den auf zwei Dezimalstellen gerundeten Wert zurück.

versionCompare()

Analysiert die Eingabeparameter, die die tatsächlichen und erwarteten Werte für eine erforderliche Eigenschaft darstellen, und vergleicht jeden Teil der Version, um festzustellen, ob der erste Wert (tatsächlich) größer als der zweite Wert (erwartet) ist.

Zweck

Diese Funktion prüft zunächst, ob die Funktion zwei Versionen als Eingabeparameter empfängt. Sie verwendet awk, um jede Version zu analysieren und in ihre Bestandteile aufzuteilen, wobei der Punkt (".") als Trennzeichen verwendet wird, um den Wert in seine Bestandteile zu unterteilen. Anschließend führt sie eine Schleife durch, um jeden Teil der ersten Version mit demselben Teil der zweiten Version zu vergleichen und festzustellen, ob die beiden Teile gleich identisch sind.

Übergeordnete Funktionen

Tabelle 43. Übergeordnete Funktionen, die versionCompare aufrufen

Übergeordnete Funktion, Script	Beschreibung
db2.home_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für den Plattenspeicherplatz für die erforderliche Eigenschaft für das DB2-Ausgangsverzeichnis.
oracle.Client_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für den Oracle-Client.
os.locale_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die Ländereinstellung des Betriebssystems.
os.MozillaVersion_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für Mozilla Firefox.
os.package.perl_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für das Perl-Paket. Sie ruft sich selbst auf.
os.RAMSize_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für den Arbeitsspeicher.
os.space_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für den verfügbaren Plattenspeicherplatz.
OS_Version_compare.sh	Vergleicht den tatsächlichen und den erwarteten Wert für die erforderliche Eigenschaft für die Version des Betriebssystems.

Syntax

```
versionCompare real expected
```

Eingabeparameter

String *$real*
Enthält den tatsächlichen Wert für eine erforderliche Eigenschaft.

String *$expected*
Enthält den erwarteten Wert für eine erforderliche Eigenschaft.

Rückgabewerte

Integer *0*
Gibt den Rückgabecode 0 zurück, wenn der tatsächliche und der erwartete Wert gleich sind. Die übergeordnete Funktion gibt "PASS" zurück.

Sonderfall: Gibt den Rückgabecode 0 zurück und endet, wenn die Funktion leere Eingabeparameter empfängt.

Integer *-1*
Gibt den Rückgabecode -1 zurück, wenn der tatsächliche Wert kleiner ist als der erwartete Wert. Die übergeordnete Funktion gibt `"FAIL"` zurück.

Gibt den Rückgabecode -1 zurück und endet, wenn die Funktion einen zweiten Eingabeparameter empfängt, der leer ist.

Integer *1*
Gibt den Rückgabecode 1 zurück, wenn der tatsächliche Wert größer ist als der erwartete Wert. Die übergeordnete Funktion gibt `"PASS"` zurück.

Gibt den Rückgabecode 1 zurück und endet, wenn die Funktion einen ersten Eingabeparameter empfängt, der leer ist.

checkHpux()

Prüft, ob die Version des Betriebssystems HP-UX für IA64- oder PARISC-Plattformen bestimmt ist.

Zweck

Diese Funktion verwendet das Flag `-m` des Befehls **uname**, um festzustellen, ob das Betriebssystem HP-UX für IA64- oder PARISC-Plattformen bestimmt ist.

Syntax

```
checkHpux
```

Rückgabewerte

String *HPUXIA64|HPUXPARISC*
Gibt `"HPUXIA64"` zurück, wenn das Flag `-m` `"ia64"` ist, andernfalls gibt sie `"HPUXPARISC"` zurück.

checkLinux()

Prüft, ob die Version des Linux-Betriebssystem für System-p-, System-z- oder x86-Plattformen bestimmt ist.

Zweck

Diese Funktion verwendet das Flag `-m` des Befehls **uname**, um festzustellen, ob das Linux-Betriebssystem für System-p-, System-z- oder x86-Plattformen bestimmt ist.

Syntax

```
checkLinux
```

Eingabeparameter

Rückgabewerte

String *LINUXPSERIES*|*LINUXZSERIES*|*LINUXX86*
Gibt "LINUXPSERIES" zurück, wenn das Flag -m "ppc64" oder "ppc" ist. Sie gibt "LINUXZSERIES" zurück, wenn der Wert "s390x" oder "s390" ist, andernfalls gibt sie "LINUXX86" zurück.

checkSunOS()

Prüft, ob die Version des Betriebssystems Solaris für SPARC- oder X86-Plattformen bestimmt ist.

Zweck

Diese Funktion verwendet das Flag -p des Befehls **uname**, um festzustellen, ob das Betriebssystem Solaris für SPARC- oder X86-Plattformen bestimmt ist.

Syntax

```
checkSunOS
```

Eingabeparameter

Rückgabewerte

String *SOLARISSPARC*|*SOLARISX86*
Gibt "SOLARISSPARC" zurück, wenn das Flag -p "sparc" ist, andernfalls gibt sie "SOLARISX86" zurück.

getValue()

Ruft den Wert für einen Schlüssel in einer angegebenen Datei ab, wenn der Schlüssel vorhanden ist.

Zweck

Syntax

```
getValue key file
```

Eingabeparameter

String *$key*
Enthält den zu setzenden Schlüssel.

String *$file*
Enthält den Namen der Datei, die den Schlüssel enthält.

setValue()

Setzt den Wert für einen Schlüssel in einer angegebenen Datei, wenn die erforderliche Eigenschaft vorhanden ist.

Syntax

```
setValue key value file
```

Eingabeparameter

String *$key*
Enthält die zu setzende erforderliche Eigenschaft.

String *$value*
Enthält den Wert für die erforderliche Eigenschaft.

String *$file*
Enthält den Namen der Datei, die die erforderliche Eigenschaft enthält.

copyValue()

Ruft den Wert für die erforderliche Eigenschaft (Schlüssel) basierend auf dem Produkt und dem Betriebssystem ab und legt diese fest.

Zweck

Diese Funktion ruft die Funktion **getValue()** auf, um den Wert für die angegebene erforderliche Eigenschaft für das Produkt und das Betriebssystem abzurufen. Anschließend ruft sie die Funktion **setValue()** auf, um den Wert für die erforderliche Eigenschaft in der Datei von Prerequisite Scanner zu setzen.

Syntax

```
copyValue key file
```

Eingabeparameter

String *$key*
Enthält den abzurufenden und zu setzenden Schlüssel.

String *$file*
Enthält den Namen der Datei, die den Schlüssel enthält.

Rückgabewerte

getSystemId()

Ruft verschiedene Betriebssystemfunktionen auf, um das relevante Betriebssystem auf den Plattformen zu suchen.

Zweck

Diese Funktion ruft verschiedene Betriebssystemfunktionen auf, um die Plattformen für das relevante Betriebssystem zu bestimmen.

Syntax

```
getSystemId
```

Eingabeparameter

Rückgabewerte

String *AIX|Linux*
Gibt "AIX" oder "Linux" zurück, wenn das Produkt Tivoli License Compliance Manager und das Betriebssystem AIX oder Linux ist, bzw. "AIX", wenn das Produkt Tivoli Asset Discovery for Distributed und das Betriebssystem AIX ist.

getClosestExistingParentDir()

Ruft das unmittelbar übergeordnete Verzeichnis oder sich selbst ab.

Zweck

Syntax

```
getClosestExistingParentDir dirpath
```

Eingabeparameter

String *$dirpath*
Enthält den Pfad zum Abrufen des übergeordneten Verzeichnisses oder den Pfad zu sich selbst.

Rückgabewerte

String *dirpath*
Gibt das übergeordnete Verzeichnis oder sich selbst zurück.

parseDirParameter()

Analysiert den Parameter aus der Parameterliste für das Flag "-p" des Scanners und speichert den Wert in der Liste.

Zweck

Syntax

Eingabeparameter

String

Rückgabewerte

printDirSize()

Überprüft den NFS-Status des angehängten Dateisystems und ruft dann den Plattenspeicherplatz des Dateisystems bzw. dessen übergeordneten Verzeichnisses ab.

Zweck

Diese Funktion ruft zuerst die Funktion **NFScheck** auf, um den NFS-Status des Verzeichnisses zu ermitteln. Wenn der Status "true" ist, ruft sie die Funktion **getClosestExistingParentDir** auf, um das Verzeichnis oder dessen übergeordnetes Verzeichnis zurückzugeben, und verwendet dann den Befehl **df**, um den freien Plattenspeicherplatz abzurufen. Abschließend ruft sie die Funktion **formatSizeDisplay** auf, um den Wert um die Dezimalstellen abzurunden.

Syntax

```
printDirSize dirpath
```

Eingabeparameter

String *$dirpath*
Enthält den Pfad zu dem Verzeichnis, für das der freie Plattenspeicherplatz abgerufen werden.

Rückgabewerte

Integer *dsize*
Gibt den freien Plattenspeicherplatz auf zwei Dezimalzeichen gerundet zurück.

String *"NFS_NOT_AVAILABLE"*
Gibt zurück, dass das angehängte Dateisystem nicht verfügbar ist.

Anhang K. Protokolldienstprogrammfunktionen für UNIX-Systeme

Prerequisite Scanner enthält eine Gruppe allgemeiner Protokollierungsfunktionen in der Datei `/lib/common_function.sh` für das Schreiben von Debug- und Tracedaten in Protokolldateien.

Tabelle 44 beschreibt die Protokolldienstprogramme.

Tabelle 44. Protokolldienstprogrammfunktionen auf UNIX-Systemen

Funktion	Beschreibung	Eingabeparameter

`wrlTrace` *`log_str1 log_str2`*	Schreibt die Zeichenfolgen *`log_str1`* und *`log_str2`* mit einer Zeitmarke in die Tracedatei.	*`log_str1`* und *`log_str2`* sind Tracezeichenfolgen, die die Aktion und den Collector darstellen, die ausgeführt und in der Tracedatei protokolliert werden. Beispiel:
		`` `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`*	Übergibt die Funktion *`fn_name`* an `wrlTrace()`.	*`fn_name`* ist die Tracezeichenfolge, die die soeben aufgerufene Funktion darstellt. Beispiel: `` `wrlTraceFuncStart "$1"` ``
`wrlTraceFuncExit` *`fn_name`*	Übergibt die Funktion *`fn_name`* an `wrlTrace()`.	*`fn_name`* ist die Tracezeichenfolge, die die soeben ausgeführte Funktion darstellt. Beispiel: `` `wrlTraceFuncExit "$1"` ``

`wrlDebug` *`log_str1 log_str2`*	Übergibt die Zeichenfolgen *`log_str1`* und *`log_str2`* an `wrlDebugGeneric()`.	*`log_str1`* und *`log_str2`* sind Debugzeichenfolgen, die die Aktion und den Collector darstellen, die ausgeführt und in der Debugdatei protokolliert werden. Beispiel:
		`` `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`*	Übergibt die Funktion *`fn_name`* an `wrlDebug()`.	*`fn_name`* ist die Debugzeichenfolge, die die soeben aufgerufene Funktion darstellt. Beispiel: `` `wrlDebugFuncStart "$1"` ``

Tabelle 44. Protokolldienstprogrammfunktionen auf UNIX-Systemen (Forts.)

Funktion	Beschreibung	Eingabeparameter
`wrlDebugFuncExit` *fn_name*	Übergibt die Funktion *fn_name* an `wrlDebug()`.	*fn_name* ist die Debugzeichenfolge, die die soeben ausgeführte Funktion darstellt. Beispiel: `` `wrlDebugFuncExit "$1"` ``
`wrlDebugFuncReturn` *result_value*	Schreibt das zurückgegebene Ergebnis *result_value* für die Funktion in die Protokolldatei.	*result_value* ist die Debugzeichenfolge, die den von der Funktion zurückgegebenen Wert darstellt. Beispiel: `` `wrlDebugFuncReturn "$versionCompare"` ``
`wrlDebugFuncParam` *param1 param2*	Übergibt die Parameter *param1* und *param2* an `wrlDebugFunc()`.	*param1* und *param2* sind die Debugzeichenfolgen, die geparste Abschnittstitel, geparste Qualifikationsmerkmale oder geparste Eingabeargumente für aufgerufene Funktionen darstellen. Beispiel: `` `wrlDebugFuncParam "OSArch" "$3"` ``
`wrlDebugGeneric formatspec` *log_str1 log_str2*	Schreibt die Zeichenfolgen *log_str1* und *log_str2*, formatiert mit dem Zeichenfolgeargument *formatspec*, in die Debugdatei.	• *log_str1* und *log_str2* sind Zeichenfolgen, die bestimmte Daten darstellen, die in einer Zeile in der Debugdatei protokolliert werden sollen. • *formatspec* ist das Zeichenfolgeargument, das hinter der Zeitangabe und links vor den Protokollzeichenfolgen und dem Zeilenvorschubzeichen eingefügt werden soll. Beispiel: `` `wrlDebugGeneric "" "$1" "$2"` ``
`wrlDebugFunc` *str*	Übergibt ein Tabulatorzeichen und den Eingabeparameter *str* an `wrlDebugGeneric()`.	*str* ist die Zeichenfolge, die die zu protokollierenden Daten darstellt, d. h. den Status einer ausgeführten Prüfung bzw. Aktion. Beispiel: `` `wrlDebugFunc "Reading config file and parsing using parse array..." ` ``
`wrlLogFuncStart` *str*	Übergibt den Eingabeparameter *str* an `wrlTraceFuncStart()` und `wrlDebugFuncStart()`.	*str* ist die Zeichenfolge, die die zu protokollierenden Daten darstellt, d. h. den Namen der aufgerufenen Funktion. Beispiel: `` `wrlLogFuncStart "main()"` ``
`wrlLogFuncExit` *str*	Übergibt den Eingabeparameter *str* an `wrlTraceFuncExit()` und `wrlDebugFuncExit()`.	*str* ist die Zeichenfolge, die die zu protokollierenden Daten darstellt, d. h. den Namen der beendeten Funktion. Beispiel: `` `wrlLogFuncExit "main()"` ``

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieser Dokumentation ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des in diesem Dokument beschriebenen Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichten Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Aussagen über Pläne und Absichten von IBM unterliegen Änderungen oder können zurückgenommen werden und repräsentieren nur die Ziele von IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufs. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren und können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

COPYRIGHTLIZENZ:

Diese Veröffentlichung enthält Beispielanwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Beispielprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten. Sie dürfen diese Beispielprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, verwenden, vermarkten oder zu verteilen, die mit den IBM Anwendungsprogrammierschnittstellen konform sind.

Wird dieses Dokument als Softcopy (Book) angezeigt, sind Fotografien oder Farbabbildungen möglicherweise nicht sichtbar.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corporation. Weitere Produkt- und Servicenamen können Marken von IBM oder anderern Herstellern sein. Eine aktuelle Liste der IBM Marken finden Sie auf der Webseite "Copyright and trademark information" unter ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript und alle auf Adobe basierenden Marken sind Marken oder eingetragene Marken der Adobe Systems Incorporated in den USA und/oder anderen Ländern.

Cell Broadband Engine und Cell/B.E. werden unter Lizenz verwendet und sind Marken der Sony Computer Entertainment, Inc. in den USA und/oder anderen Ländern.

Intel, das Intel-Logo, Intel Inside, das Intel Inside-Logo, Intel Centrino, das Intel Centrino-Logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium und Pentium sind Marken oder eingetragene Marken der Intel Corporation oder ihrer Tochtergesellschaften in den USA oder anderen Ländern.

IT Infrastructure Library ist eine eingetragene Marke der Central Computer and Telecommunications Agency. Die Central Computer and Telecommunications Agency ist nunmehr in das Office of Government Commerce eingegliedert worden.

ITIL ist eine eingetragene Marke, eine eingetragene Gemeinschaftsmarke des Cabinet Office und eine eingetragene Marke, die beim US Patent and Trademark Office registriert ist.

Linux ist eine Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Java und alle Java-basierten Marken und Logos sind registrierte Marken von Sun Microsystems, Inc. und den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Unterstützungsinformationen und Feedback

Wenn bei der Verwendung Ihrer IBM Software Probleme auftreten, möchten Sie diese schnell lösen. IBM stellt verschiedene Möglichkeiten bereit, die benötigte Unterstützung zu erhalten. Dies kann zum Beispiel online oder über IBM Support Assistant geschehen. Sie können auch Feedback oder Produktanforderungen für Erweiterungen einreichen.

Online

Die folgenden Sites enthalten Informationen zur Fehlerbehebung:

- Rufen Sie die Webseite zu IBM Prerequisite Scanner in IBM Support Portal auf.
- Sehen Sie sich die Artikel zu Prerequisite Scanner in Service Management Connect an. Sie können auch Beiträge zu diesen Artikeln schreiben.

Verwenden Sie die folgenden Sites, um Feeback zu geben, Anforderungen einzureichen oder über Prerequisite Scanner zu diskutieren:

- Sehen Sie sich die Artikel zu Prerequisite Scanner auf der Website Prerequisite Scanner at Service Management Connect an. Sie können Beiträge zu diesen Artikeln schreiben.
- Verwenden Sie Integrated Service Management Message Board auf Service Management Connect.
- Prüfen und reichen Sie Produkterweiterungsanforderungen für Prerequisite Scanner in der Tivoli-RFE-Community ein.

IBM Support Assistant

IBM Support Assistant (ISA) ist eine kostenlose lokale Workbench für die Wartungsfreundlichkeit Ihrer Software, die Ihnen hilft, Fragen zu IBM Softwareprodukten zu klären und auftretende Probleme zu lösen. ISA bietet einen schnellen Zugriff auf unterstützungsrelevante Informationen und auf Tools, die der Servicefreundlichkeit dienen, um Fehler zu ermitteln. Rufen Sie zum Installieren der ISA-Software die Website unter der Adresse http://www.ibm.com/software/support/isa auf.

Index

H

I

K

M

N

O

P

Q

R

S

T

U

V

W

X

Gedruckt in Deutschland