# Guide Sun™ SNMP Management Agent pour Sun Fire™ B1600

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054 États-Unis
650-960-1300

Référence n° 817-2500-10
Avril 2003, révision 01

Envoyez vos commentaires sur ce document à l'adresse suivante : `http://www.sun.com/hwdocs/feedback`

# Table des matières

# Figures

# Tableaux

# Exemples de code

# Préface

Ce guide est une description de Sun SNMP Management Agent pour la plate-forme Sun Fire B1600, logiciel assurant la gestion du matériel de la plate-forme à l'aide du protocole Simple Network Management Protocol (SNMP).

Ce logiciel assure la *surveillance* de l'inventaire, la configuration, et la création de rapports d'erreurs et liés à l'environnement. Il permet également de *contrôler* et de *surveiller* les indicateurs de services, l'alimentation, la mise en attente et la réinitialisation des lames du processeur.

Il se destine à des administrateurs d'entreprise et des développeurs professionnels chevronnés.

Ce guide est divisé en deux parties :

- La première partie (du Chapitre 1 au Chapitre 8) se veut une introduction à SNMP Management Agent et décrit ses fonctions.
- La seconde partie (du Chapitre 9 au Chapitre 13) explique comment installer et configurer le logiciel.

## Organisation de ce guide

Ce guide contient les chapitres suivants :

**Première partie**

Le **Chapitre 1** décrit les composants du logiciel Sun SNMP Management Agent.

Le **Chapitre 2** contient une brève description des fonctions principales du protocole Simple Network Management Protocol (SNMP).

Le **Chapitre 3** décrit l'usage et les fonctionnalités de l'agent maître SNMPv3.

Le **Chapitre 4** contient une présentation de la modélisation de Sun Fire B1600 par SNMP.

Le **Chapitre 5** décrit comment les objets gérés de Sun Fire B1600 et leurs relations sont présentés par l'interface SNMP.

Le **Chapitre 6** décrit la hiérarchie des classes physiques sunPlat et le mode de représentation des classes d'objets physiques gérées définies dans le modèle sunPlat par la base SUN-PLATFORM-MIB.

Le **Chapitre 7** décrit la hiérarchie des classes logiques sunPlat et le mode de représentation des classes d'objets gérées définies dans le modèle sunPlat par la base SUN-PLATFORM-MIB.

Le **Chapitre 8** décrit les classes et les attributs des notifications SunPlat, définies dans la base d'informations SUN-PLATFORM-MIB.

**Seconde partie**

Le **Chapitre 9** décrit les composants du logiciel de gestion de la plate-forme Sun Fire B1600 et dresse la liste des vérifications nécessaires du système avant de procéder à l'installation du logiciel SNMP.

Le **Chapitre 10** décrit comment installer le logiciel de gestion sur la plate-forme Sun Fire B1600.

Le **Chapitre 11** contient des informations sur les fichiers de configuration utilisateur.

Le **Chapitre 12** décrit la configuration par défaut après l'installation et explique comment modifier les fichiers de configuration.

Le **Chapitre 13** explique comment désinstaller le logiciel.

Le **Chapitre 14** fournit une aide au dépannage du logiciel.

L'**annexe A** décrit comment installer J2RE pour une utilisation conjointe avec J2SE, et comment modifier les scripts de démarrage pour localiser l'installation.

# Conventions typographiques

| Type de caractère ou symbole | Signification | Exemples |
|---|---|---|
| `AaBbCc123` | Noms de commandes, de fichiers et de répertoires ; affichage sur -l'écran de l'ordinateur. | Modifiez le fichier `.login`. Utilisez `ls -a` pour dresser la liste de tous les fichiers.<br>`% You have mail.` |
| **`AaBbCc123`** | Ce que vous saisissez, par opposition à l'affichage sur l'écran de l'ordinateur. | `% su`<br>`Password:` |
| *AaBbCc123* | Titres d'ouvrages, nouveaux mots ou termes, mots importants. Remplace les variables de ligne de commandes par des noms ou des valeurs réels. | Lisez le chapitre 6 du *Guide de l'Utilisateur.*<br>Il s'agit d'options de *classe.*<br>Pour supprimer un fichier, tapez **`rm`** *nom de fichier.* |

# Invites de shell

| Shell | Invite |
|---|---|
| C shell | *machine-name*% |
| Super-utilisateur du C shell | *machine-name*# |
| Bourne shell et Korn shell | $ |
| Super-utilisateur du Bourne shell et du Korn shell | # |

# Documentation connexe

| Application | Titre | Référence produit |
|---|---|---|
| SunMC | *SunMC 3.0 Supplement for the Sun Fire B1600* | 817-1011 |
| Notes de version | *SNMP Release Notes for the Sun Fire B1600* | 817-1006 |
| Plate-forme Sun Fire B1600 | *Sun Fire B1600 Blade System Chassis Hardware Installation Guide* | 816-7614 |
| | *Sun Fire B1600 Blade System Chassis Software Setup Guide* | 816-3361 |
| | *Sun Fire B1600 Blade System Chassis Administration Guide* | 816-4765 |
| | *Sun Fire B1600 Blade System Chassis Switch Administration Guide* | 816-3365 |

# Accès à la documentation Sun

Vous pouvez afficher, imprimer ou acquérir une vaste sélection de documents Sun, notamment de versions localisées, à l'adresse suivante :

`http://www.sun.com/documentation`

# Prise de contact avec l'équipe de support technique

Si vous avez des questions techniques sur ce produit qui ne sont pas abordées dans ce document, consultez la page Web suivante :

`http://www.sun.com/service/contacting`

# Vos commentaires sont les bienvenus

Dans le souci d'améliorer notre documentation, nous vous invitons à nous faire parvenir vos commentaires et vos suggestions. Vous pouvez soumettre vos commentaires à l'adresse suivante :

`http://www.sun.com/hwdocs/feedback`

Veuillez inclure le titre et le numéro de référence du document dans vos commentaires :

*Guide de Sun SNMP Management Agent pour Sun Fire B1600*, numéro de référence 817-2500-10.

# PIÈCE I Description technique et fonctionnalités

CHAPITRE 1

# Complément de Sun SNMP Management Agent

Cette version du logiciel Sun™ SNMP Management Agent assure la surveillance et le contrôle de la plate-forme Sun Fire ™ B1600 et du serveur-lame Sun Fire B100s.

En fonction du type de plate-forme, vous pouvez utiliser :

- Un agent domaine, exécuté sur le serveur-lame Sun Fire B100s (surveillance de domaine matérielle)

  Le logiciel est installé localement sur le serveur surveillé et seul ce serveur peut être analysé. Dans le cas de la plate-forme Sun Fire B1600, chaque lame est surveillée séparément.

  La surveillance du domaine matérielle pour le serveur-lame Sun Fire B100s se limite au matériel de la lame. Elle n'inclut pas d'autres composants de la plate-forme, tels que les indicateurs de services, les PSU, les SSC et l'identité même de la plate-forme.

- Un agent plate-forme, mandaté par un contrôleur de système (surveillance de plate-forme matérielle)

  Le logiciel est installé sur un serveur (agent plate-forme) distant qui accède à l'instrumentation de la plate-forme par le biais du contrôleur de système. Ceci vous permet de surveiller tout le matériel géré par le contrôleur du système.

  La surveillance de la plate-forme matérielle de Sun Fire B1600 inclut l'étagère, son identité, les indicateurs de services et toutes ses unités remplaçables par l'utilisateur. En outre, elle donne accès à certaines informations matérielles (surveillance de la tension notamment) sur les lames Sun Fire B100s, contrairement à la surveillance du domaine matérielle.

  La FIGURE 1-1 est un exemple de deux types de surveillance matérielle. Les étagères A et B de Sun Fire B1600 sont connectées au poste de gestion du réseau via un serveur agent plate-forme (surveillance de la plate-forme matérielle). Dans le cas de l'étagère C de Sun Fire B1600, les lames Sun Fire B100s sont directement connectées au poste de gestion du réseau (surveillance du domaine matérielle).

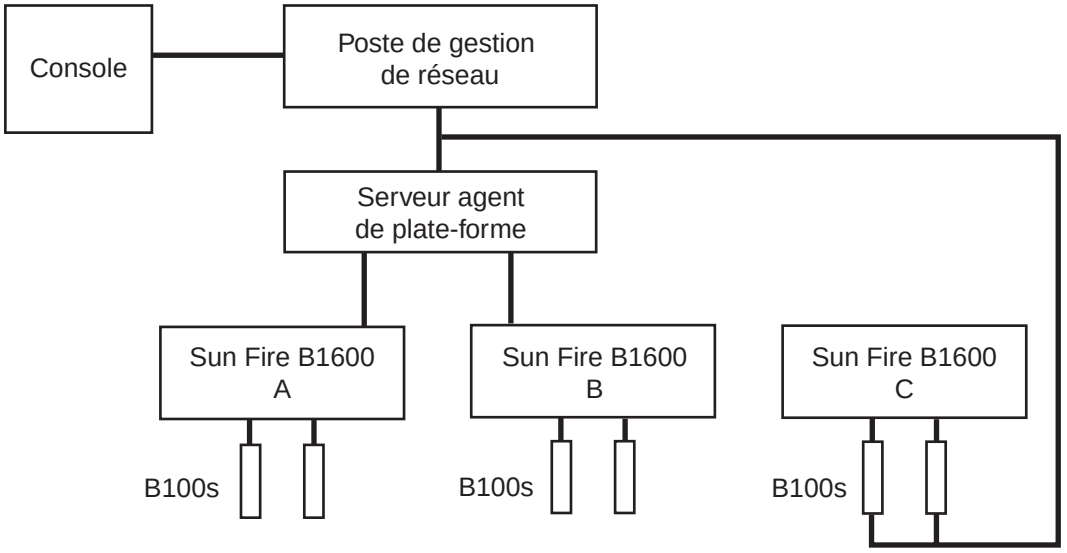


**FIGURE 1-1** Exemple de surveillance du domaine et de la plate-forme matérielle

Le logiciel comprend un nombre de modules offrant les fonctionnalités suivantes :

- Un sous-agent SNMP

  Par défaut, le sous-agent SNMP est enregistré comme sous-agent de l'agent maître Solaris, snmpdx. Le sous-agent est aussi désigné comme *médiateur SNMP*.

- Un agent maître SNMPv3

  L'agent maître SNMPv3 offre un point de présence unique et sécurisé à partir desquels vous pouvez accéder aux médiateurs SNMP résidant sur la plate-forme. L'agent maître agit comme un proxy en envoyant ses requêtes à `snmpdx`.

- Sun Fire B1600 et à l'instrumentation de Sun Fire B100s.

  Ces modules sont installés selon les besoins, selon que la surveillance du domaine ou de la plate-forme est utilisée.

CHAPITRE 2

# Introduction au protocole SNMP

Ce chapitre contient une brève introduction aux principales fonctionnalités du protocole Simple Network Management Protocol (SNMP). Il ne constitue en aucun cas une analyse détaillée et répond aux problèmes spécifiques au système Sun Fire™ B1600.

Le chapitre contient les sections suivantes :



## Versions du SNMP

SNMP est une norme Internet standard permettant de gérer des périphériques réseau (systèmes). Il est défini, en accord avec d'autres normes Internet, par un nombre de demandes de commentaires (Requests for Comments ou RFC) publiées par la Internet Engineering Task Force (IETF).

Il existe trois versions de protocoles SNMP définissant les normes approuvées :

- SNMPv1
- SNMPv2 (également connue et désignée dans ce document sous le nom SNMPv2c)
- SNMPv3

SNMPv1 a été pour la première fois défini en 1988. SNMPv2 a été créé en 1993 en vue de pallier à certaines limitations du protocole SNMPv1 en incluant des opérations de protocole et des types de données supplémentaires et en assurant une meilleure sécurité. Les limitations du modèle de sécurité ont mené à la création maintenant reconnue de la norme SNMPv2c, qui inclut de nouvelles fonctions en matière de sécurité. Des versions test, connues sous le nom de SNMPv2usec et SNMPv2* sont également apparues, mais n'ont pas été adoptées à grande échelle et demeurent à l'état expérimental.

Créée en 1999, la norme SNMPv3 définit le cadre de gestion SNMP prenant en charge des composants enfichables, notamment des fonctions de sécurité.

Pour des informations détaillées sur ces normes, reportez-vous aux RFC suivants sur le site Web de la IETF (`http://www.ietf.org/rfc.html`) :

- SNMPv1 : RFC1155, RFC1157, RFC1212, RFC1215
- SNMPv2 : RFC2578, RFC2579, RFC2580, RFC3416
- SNMPv3 : RFC3410, RFC3411, RFC3412, RFC3413, RFC3414, RFC3415
- Co-existence entre les normes : RFC2576

# Gestionnaires et agents SNMP

SNMP est un protocole réseau permettant à des périphériques d'être gérés à distance par un poste de gestion réseau (Network Management Station ou NMS), communément appelé un *gestionnaire*.

Pour être géré, un périphérique doit être associé à un *agent SNMP* (appelé le médiateur SNMP). L'objectif du médiateur est de :

- recevoir du gestionnaire des demandes de données représentant l'état du périphérique et fournir une réponse appropriée ;
- accepter des données du gestionnaire pour permettre de contrôler l'état du périphérique ;
- générer des traps *SNMP*, messages indésirables envoyés à un ou plusieurs gestionnaires sélectionnés pour signaler des événements significatifs concernant le périphérique.

# Base d'informations de gestion SNMP

Pour gérer et surveiller un périphérique, ses caractéristiques doivent être représentées dans un format connu de l'agent et du gestionnaire. Ces caractéristiques peuvent représenter des propriétés physiques, telles que les vitesse du ventilateurs, ou des services, tels que les tables de routage. La structure des données définissant ces caractéristiques s'appelle une *base d'informations de gestion ou Management Information Base* (MIB). Ce modèle de données se présente généralement sous forme de tables, mais peut également inclure de simples valeurs. La table de routage est un exemple de table et l'horodateur indiquant l'heure de démarrage de l'agent est un exemple de valeur simple.

La MIB est une définition de magasin de données virtuel accessible via SNMP. Son contenu est accessible depuis le gestionnaire à l'aide des opérations `get` et `set` suivantes :

- En réponse à une opération `get`, le médiateur fournit des données, gérées localement ou directement depuis le périphérique géré.
- En réponse à une opération `set`, l'agent effectue généralement une action affectant son propre état ou celui du périphérique géré.

Pour permettre à un NMS de gérer un périphérique via son agent, la MIB correspondant aux données présentées par l'agent doit être chargée dans le gestionnaire. Le mécanisme permettant d'effectuer cette opération varie en fonction de l'implémentation du logiciel de gestion réseau. Cette opération fournit au gestionnaire les informations nécessaires pour traiter et interpréter correctement le modèle de données présenté par l'agent.

---

**Remarque –** Les bases d'informations MIB peuvent faire référence à des définitions contenues dans d'autres bases. Ainsi, pour utiliser une MIB donnée, il peut être nécessaire d'en charger d'autres.

---

Pour traiter le contenu de ce magasin de données virtuelles, la MIB est définie en termes d'*identificateurs d'objets* (IDO). Un IDO se compose d'une série d'entiers ordonnés par ordre hiérarchique qui définit un espace de nom unique. Chaque entier affecté est associé à un nom textuel. Par exemple, l'IDO `1.3.6.1` correspond au nom de l'IDO `iso.org.dod.internet` et `1.3.6.1.4` correspond au nom de l'IDO `iso.org.dod.internet.private`.

La forme numérique est utilisée dans les transactions du protocole SNMP, tandis que la forme textuelle est utilisée dans des interfaces utilisateur pour faciliter la lecture. Les objets représentés par ces IDO sont fréquemment désignés par le dernier composant de leur nom sous forme sténographique. Pour éviter toute confusion émanant de cette convention, il est courant d'appliquer un préfixe spécifique à la MIB, tel que *sunPlat*, à tous les noms d'objets qui y sont définis. Ces identificateurs sont ainsi globalement uniques.

**Remarque –** La MIB est définie à l'aide d'un langage appelé ASN.1, dont les principes font l'objet d'un document distinct. À titre de référence, la structure des MIB pour SNMPv2c est définie par sa structure d'informations de gestion ou Structure of Management Information (SMI) dans RFC2578. Cette structure définit la syntaxe et les types de données de base accessibles aux MIB. Les conventions textuelles (définitions du type) contenues dans RFC2579 définissent des types de données et les énumérations supplémentaires.

## Tables MIB

La majeure partie du contenu des données définie par les MIB se présente sous forme de tableau et est organisée par des entrées comprenant une série d'objets, chacun doté de son propre IDO. Par exemple, une table des caractéristiques du ventilateur se compose d'un nombre de lignes (un par ventilateur), chaque ligne contenant des colonnes correspondant à la vitesse actuelle, la vitesse escomptée et la vitesse minimum acceptable.

Le traitement des lignes de la table peut être :

- un index unidimensionnel simple (un numéro de ligne dans la table, par exemple 6) ;
- un indicateur d'instance multidimensionnelle plus complexe, comme une adresse IP et un numéro de port (par exemple, `127.0.0.1, 1234`).

Chaque définition de table au sein de la MIB inclut une clause INDEX qui définit quels indicateurs d'instances doivent être utilisés pour sélectionner une entrée donnée. Dans tous les cas, les objets permettant de définir l'index dans la ligne requise doivent être eux-mêmes définis dans la MIB. Ainsi, une table comportant un index unidimensionnel simple inclut généralement une colonne d'index référencée par la clause INDEX de la table. Un élément de données spécifique dans une table est ensuite traité en spécifiant l'IDO contenant son préfixe de colonne.

Par exemple, l'élément `myFanTable.myFanEntry.myCurrentFanSpeed` incluant un indicateur d'instance à suffixe donne `myFanTable.myFanEntry.myCurrentFanSpeed.127.0.0.1.1234`.

Le SMI définissant la syntaxe de la MIB offre une puissance d'extension des tables puisqu'elle permet d'ajouter des entrées de manière efficace en ajoutant des colonnes supplémentaires à la table. Pour cela, il convient de définir une table avec une clause INDEX qui soit la version dupliquée de la clause INDEX de la table en cours d'extension.

Il est également possible de définir des tables MIB qui sont indexées, non par des objets qu'elles contiennent, mais par des objets *importés* d'autres tables, potentiellement définis dans d'autres MIB. Cette construction permet aux colonnes d'êtreajoutées de manière efficace à une table existante.

---

**Remarque –** La base SUN-PLATFORM-MIB utilise fréquemment ce mécanisme pour étendre des tables définies dans la ENTITY-MIB (voir Chapitre 5).

---

# Contrôle de l'accès

Tous les objets traitables définis dans la MIB sont associés à des droits d'accès maximum, *lecture seule* ou *lecture-écriture*. Ces derniers déterminent l'accès maximal pris en charge par l'agent et peuvent être utilisés par le gestionnaire pour empêcher les utilisateurs d'effectuer certaines opérations. L'agent est à même d'appliquer des droits d'accès inférieurs selon les besoins ; en effet, il peut refuser des droits d'écriture sur des objets pour lesquels des droits en lecture-écriture sont définis. Ce refus peut être décidé en fonction :

- de l'applicabilité de l'opération à l'objet actuellement traité (par exemple, cas d'un objet défini par la MIB représentant la machine d'état pour laquelle seules certaines transactions sont valides) ;
- des restrictions de sécurité limitant certaines opérations à un groupe restreint de gestionnaires.

Le mécanisme utilisé pour communiquer des droits d'accès de sécurité dans SMMPv1 s'appelle les *chaînes de communauté*. Ces dernières constituent de simples chaînes textuelles, telles que *private* et *public*, transmises avec chaque demande de données SNMP. Étant donné que les demandes SNMPv1 et SNMPv2 ne sont pas cryptées, cette opération n'est pas considérée comme sûre. Le mécanisme permettant de définir les chaînes de communauté auxquelles l'agent doit répondre, et de quel gestionnaire elles proviennent, dépend de la configuration de l'agent, mais est généralement basée sur des listes de contrôle d'accès ou Access Control Lists (ACLs), fichiers décrivant les autorisations d'accès applicables.

Pour savoir comment configurer des ACL, reportez-vous au Chapitre 11.

# Agents maîtres SNMP

Un gestionnaire communique avec l'agent en envoyant des paquets (UDP) via un port standard (161) au système sur lequel est exécuté l'agent. Si plusieurs agents gérant des périphériques différents sont exécutés sur un système donné, un conflit potentiel dans l'utilisation des ressources du port peut avoir lieu. Pour remédier à ce problème, vous pouvez utiliser des numéros de port différents non standard pour chaque agent. Vous pouvez également introduire le concept d'un *agent maître*, qui accepte des requêtes du SNMP au nom de tous les agents exécutés sur un système donné et qui les transmet le cas échéant. Cette opération permet au gestionnaire d'accéder de manière cohérente à tous les agents du SNMP. La plate-forme Sun Fire B1600 prend en charge les deux approches.

Pour de plus amples informations sur l'agent maître, reportez-vous au Chapitre 3.

## Médiateur SNMP et snmpdx

`snmpdx` est l'agent SNMP standard de Solaris™ et un agent maître de SNMPv1.

Par défaut, le médiateur SNMP est enregistré comme un sous-agent de `snmpdx`. Dans cette configuration, seules les requêtes `get` et `set` de SNMPv1 sont prises en charge, bien que les notifications SNMPv2c soient émises.

La relation entre `snmpdx` et le médiateur SNMP est abordée de manière plus détaillée au Chapitre 10 et au Chapitre 11.

Voir également la page man `snmpdx`(1M).

CHAPITRE 3

# Agent maître

Ce chapitre décrit l'usage et les fonctionnalités de l'agent maître SNMPv3.

Il contient les sections suivantes :



## Avantages

L'agent maître SNMPv3 offre un point de présence unique et sécurisé à partir duquel il est possible d'accéder aux informations de gestion SNMP.

L'agent maître SNMPv3 est lié au port du service SNMP (port 161 par défaut) et transmet toutes les requêtes à `snmpdx`, l'agent maître Solaris standard, qui à son tour les envoie aux sous-agents enregistrés adéquats. `snmpdx` est inclus dans le système Solaris standard, mais ne prend en charge que SNMPv1 et par conséquent, n'offre pas directement le même niveau de sécurité que SNMPv3. L'agent maître convertit toutes les requêtes, qu'elles soient SNMPv1, v2c ou v3, en SNMPv1, afin que la commande `snmpdx` puisse les traiter.

En effet, l'agent maître agit comme un pare-feu SNMP en offrant un accès sécurisé à tous les sous-agents existants.

# Présentation de la configuration

Cette section contient une introduction au mode de configuration de SNMP pour inclure la fonction Agent maître. La rubrique est abordée de manière détaillée au Chapitre 11, qui inclut une description complète des fichiers de configuration cités en référence dans le reste de cette section.

Le médiateur SNMP est enregistré comme sous-agent de snmpdx à l'aide d'un numéro de port affecté automatiquement.

---

**Remarque –** L'agent SNMP est désigné comme médiateur SNMP dans ce guide.

---

Lorsque l'agent maître est activé, le lancement automatique de la commande `snmpdx` est désactivé et l'agent maître est enregistré sur le port 161. Un nouveau numéro de port est affecté à `snmpdx`, lancée sous le contrôle du fichier de démarrage de l'agent maître.

La configuration est effectuée au moyen :

- desfichiers de sécurité SNMPv3 `spama.uacl` et `spama.security` ;
- du fichier liste de contrôle d'accès (ACL) à SNMPv1/v2 `spama.acl` ;
- du fichier de configuration `spama.conf` ;
- du script de démarrage, `spama`.

Il n'est pas possible de configurer l'agent maître de manière dynamique lors de son exécution.

Le médiateur SNMP requiert également une configuration, décrite au Chapitre 11.

CHAPITRE 4

# Le modèle de gestion de la plate-forme

Ce chapitre contient une présentation de la modélisation du système Sun Fire B1600 par SNMP à l'aide du Modèle Sun Platform SNMP (sunPlat).

Il contient les sections suivantes :



## Modélisation de la Plate-forme Sun Fire B1600

La plate-forme Sun Fire B1600 constitue un ensemble de ressources matérielles imbriquées dans un châssis. Certaines ressources peuvent être directement imbriquées dans le châssis, telles qu'une carte mère. D'autres sont imbriquées dans d'autres ressources. Par exemple, une carte mère peut inclure un processeur. Ces relations, s'étendant depuis le châssis, forment une *hiérarchie* de ressources matérielles, chacune contenues physiquement dans son *parent* affilié. Cette hiérarchie est modélisée à l'aide de *relations* entre *des objets gérés* représentant les ressources matérielles.

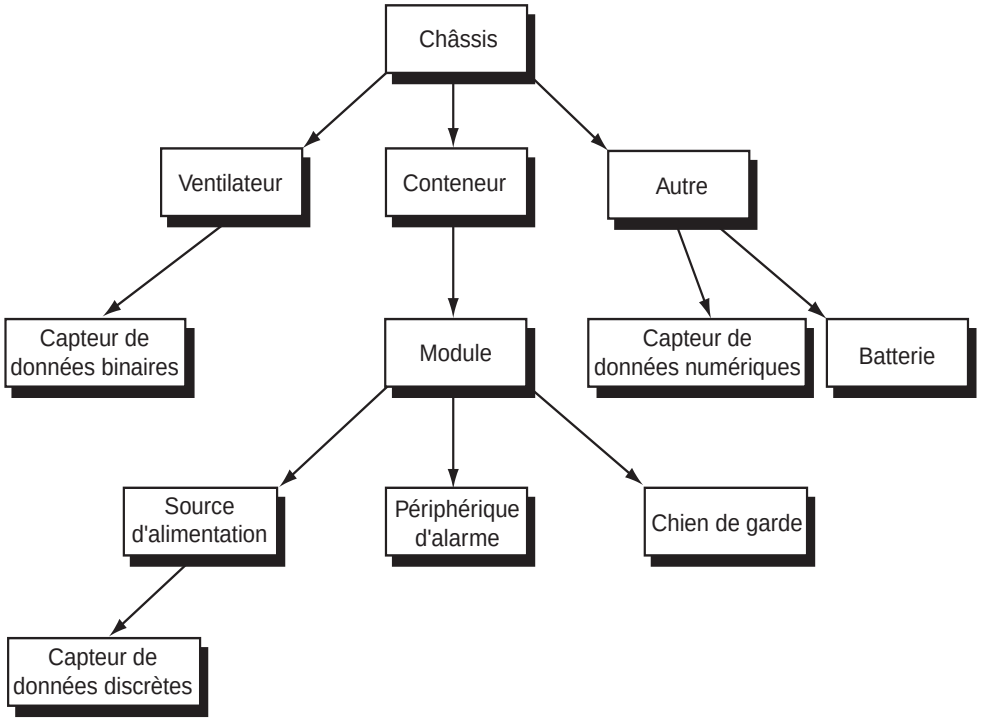


**FIGURE 4-1** Exemple de hiérarchie de ressources matérielles

# Objets gérés

Le modèle sunPlat contient un ensemble utile de blocs de construction de plate-forme communs représentant des ressources matérielles essentielles. Les instances de ces blocs de construction de plate-forme s'appellent des *objets gérés*. Une ressource matérielle est représentée par un objet géré si elle peut être surveillée ou si elle fournit des informations de configuration utiles.

D'autres objets gérés permettent de représenter d'autres fonctions de l'interface de gestion. Par exemple, les ressources matérielles peuvent émettre des rapports d'état asynchrones, (*notifications*) pour résoudre des problèmes (*alarmes*) ou suite à des modifications de la configuration (*événements*).

Les objets gérés sont définis en termes de *classes d'objets gérés*. Les caractéristiques de la ressource sont représentées par des *propriétés* de l'objet géré. De nouvelles classes, appelées *sous-classes*, sont définies en termes de classes existantes. Une sous-classe *hérite* de toutes les caractéristiques de sa *super-classe*, mais représente ses propres caractéristiques en ajoutant de nouvelles propriétés.

La FIGURE 4-2 illustre la hiérarchie de l'héritage de classes des blocs de construction matérielle définie par le modèle sunPlat.

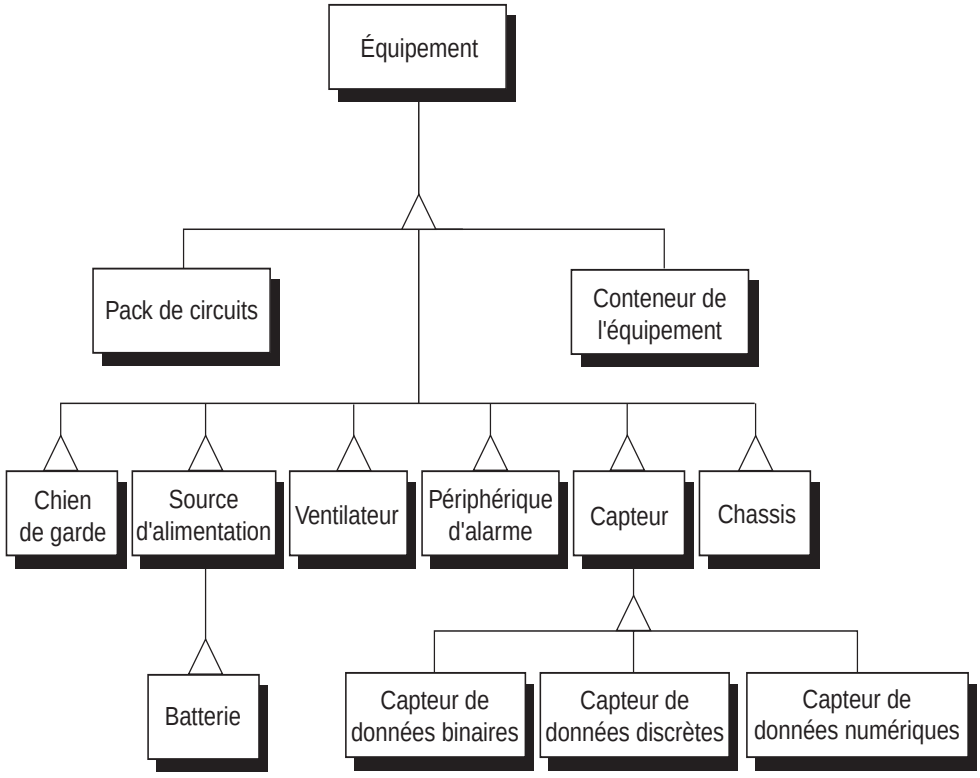


**FIGURE 4-2** sunPlat Schéma de l'héritage des classes d'objets gérés

# Dérivé des classes sunPlat

Les classes sunPlat sont basées sur des concepts de gestion standard. Le système Sun Fire B1600 utilise un sous-ensemble du modèle d'information réseau générique ITU-T, choisi pour sa représentation de l'infrastructure matérielle. Ce dernier crée un cadre puissant et extensible assurant une gestion uniforme de la configuration et des erreurs dans un réseau de gestion des télécommunications (TMN ou Telecommunications Management Network).

Le schéma du modèle d'informations commun (CIM) de la Distributed Management Task Force (DMTF) modélise l'environnement physique, ainsi que la définition et le traitement des événements, et fournit au modèle commun des extensions propres au système.

CHAPITRE **5**

# Les Sun Fire B1600 MIB

Ce chapitre décrit comment les objets gérés de la plate-forme Sun Fire B1600 et leurs relations sont présentés par l'interface SNMP.

Le chapitre contient les sections suivantes :



## Représentation du modèle dans SNMP

Le médiateur SNMP prend en charge la gestion basée sur des interrogations et des événements. Les composants physiques du système Sun Fire B1600 et une représentation logique des domaines administratifs qu'il contient sont contenus dans la base ENTITY-MIB définie par RFC 2737, étendu par la base SUN-PLATFORM-MIB.

**Remarque –** La plupart des objets définis dans les MIB ont des `DROITS D'ACCÈS MAX.` en `lecture-écriture`, mais l'écriture sur ces objets n'est possible que si elle est adaptée au composant modélisé.

La base ENTITY-MIB contient les groupes suivants, qui décrivent les éléments physiques et logiques du système géré :

### *Groupe entityPhysical*

Le groupe `entityPhysical` décrit les entités physiques, ressources physiques identifiables gérées par l'agent (par exemple, le châssis, les prises électriques, les capteurs, etc.). Ces entités sont représentées par des lignes dans la table *entPhysicalTable*.

### *Groupe entityLogical*

Le groupe `entityLogical` décrit les entités logiques gérées par l'agent. Ces dernières sont des représentations des entités logiques à *valeur élevée* assurant l'abstraction du service qui doit être géré par des niveaux de gestion supérieurs. Ces niveaux concernent principalement la gestion de la plate-forme matérielle et incluent des fonctions, telles que le redémarrage du système d'exploitation, la réinitialisation du matériel et le contrôle de l'alimentation. Généralement, ils correspondent à des domaines administratifs, tels que les domaines Solaris ou les contrôleurs de services.

### *Groupe entityMapping*

Le groupe `entityMapping` identifie la relation entre les groupes `entityPhysical` et `entityLogical`. Cette fonction est traitée internement par le médiateur SNMP.

### *Groupe entityGeneral*

Le groupe `entityGeneral` contient l'horodateur de la dernière modification lorsqu'une entité de la table Entité physique ou la table de mappage physique est modifiée.

### *Groupe entityMIBTraps*

Le groupe `entityMIBTraps` définit les notifications *entPhysicalChange* utilisées pour signaler un changement dans un objet de la base ENTITY-MIB.

Le Chapitre 2 contient une vue d'ensemble du mode de représentation du Modèle Sun Platform SNMP par les éléments génériques de SNMP.

# Le modèle physique

Le modèle physique sunPlat utilise la base ENTITY-MIB pour proposer une hiérarchie de confinement des entités matérielles. Chaque entité est modélisée comme une ligne distincte de la table entPhysicalTable de la base ENTITY-MIB.

La FIGURE 5-1 illustre un exemple de hiérarchie de confinement physique. Le numéro situé dans le coin inférieur droit indique l'index de la ligne correspondante dans la table *entPhysicalTable* (voir TABLEAU 5-1).

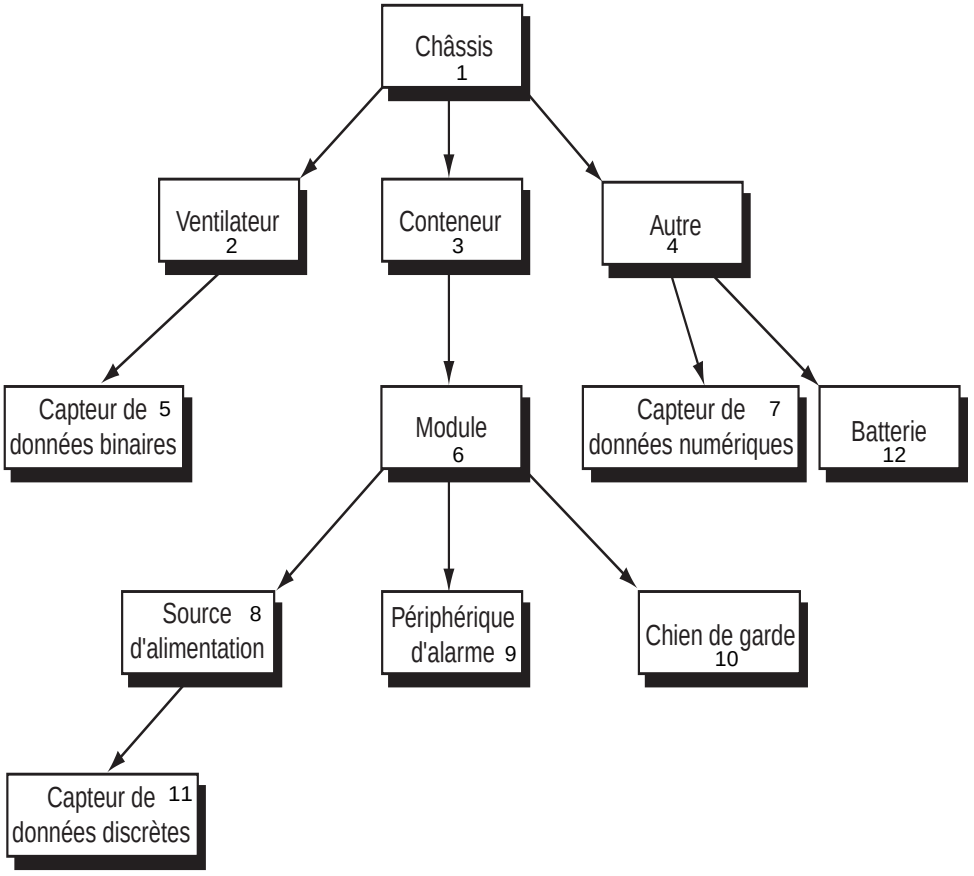


**FIGURE 5-1** Exemple de hiérarchie de ressources matérielles

Ces informations sont présentées à l'aide de tables SNMP :

- Table Entité physique (*entPhysicalTable*)

  Cette table contient une ligne pour chaque entité matérielle. Ces lignes s'appellent des *entrées* et une ligne particulière s'appelle une *instance*. Chaque entrée contient :

  - La classe physique (*entPhysicalClass*)
  - Caractéristiques communes à l'entité matérielle
  - Un index unique (*entPhysicalIndex*)
  - Une référence (*entPhysicalContainedIn*) qui pointe vers la ligne de l'entité matérielle servant de *conteneur* de cette ressource. Cette référence est nulle pour les composants, tels que les châssis, qui ne font pas physiquement partie d'un autre conteneur.

- Table de mappage physique (*entPhysicalContainsTable*)

  Cette table contient une copie virtuelle de la hiérarchie des ressources matérielles représentées dans la table Entité physique. Cette table est bidimensionnelle, indexée d'abord par l'élément *entPhysicalIndex* de l'entrée contenante, et ensuite par l'élément *entPhysicalIndex* de chaque entrée contenue.

La TABLEAU 5-1 indique la variable *entPhysicalTable* sur laquelle est basée le chiffre ci-dessus, et la TABLEAU 5-2 indique le mappage physique.

**TABLEAU 5-1** Table Entité physique

| entPhysicalIndex | entPhysicalClass | entPhysicalContainedIn |
|---|---|---|
| 1 | châssis | 0 |
| 2 | ventilateur | 1 |
| 3 | conteneur (par exemple, un emplacement contenant une unité remplaçable par l'utilisateur) | 1 |
| 4 | autre | 1 |
| 5 | capteur (binaire) | 2 |
| 6 | module (par exemple, une unité remplaçable par l'utilisateur enfichable) | 3 |
| 7 | capteur (numérique) | 4 |
| 8 | source d'alimentation | 6 |
| 9 | périphérique d'alarme | 6 |
| 10 | chien de garde | 6 |
| 11 | capteur (valeur discrète) | 8 |
| 12 | source d'alimentation (batterie) | 4 |

**TABLEAU 5-2** Table de mappage physique

| entPhysicalIndex | entPhysicalChildIndex |
|---|---|
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |
| 2 | 5 |
| 3 | 6 |
| 4 | 7 |
| 4 | 12 |
| 6 | 8 |
| 6 | 9 |
| 6 | 10 |
| 8 | 11 |

# Classes

Classe `entPhysicalClass` dans une valeur numérotée qui fournit une indication du type de matériel général d'une entité physique particulière, chacune étant représentée par une ligne dans la table *entPhysicalTable*.

Les énumérations suivantes s'appliquent à la plate-forme Sun Fire B1600 (voir également FIGURE 5-1 et TABLEAU 5-1) :

- `other(1)`

  L'énumération `other` s'applique si l'entité physique ne peut pas être classée par l'un des éléments suivants.

- `chassis(3)`

  La classe `chassis` représente un conteneur général de l'équipement. Toute classe d'entités physiques peut être contenue dans un châssis.

- `container(5)`

  La classe `container` s'applique à une entité physique pouvant contenir une ou plusieurs entités physiques amovibles, de même type ou de type différent. Par exemple, chaque emplacement vide ou occupé d'un châssis est modélisé comme un conteneur. Les unités remplaçables par l'utilisateur, telles qu'une source d'alimentation ou un ventilateur, sont modélisées comme des modules dans une entité de conteneur.

- `powerSupply(6)`

  La classe `power supply` s'applique à un composant pouvant fournir une source d'alimentation électrique.

- `fan(7)`

  La classe `fan` s'applique si l'entité physique est un ventilateur ou un autre périphérique de refroidissement.

- `sensor(8)`

  La classe `sensor` s'applique à une entité physique capable de mesurer une propriété physique.

- `module(9)`

  La classe `module` s'applique à un sous-système autonome et est modélisée dans une autre entité physique, telle qu'un `châssis` ou un autre `module`. L'entité est toujours modélisée dans un `conteneur`

# Le modèle logique

Le modèle logique sunPlat utilise la base ENTITY-MIB pour fournir une liste des entités logiques à valeur élevée. Chaque entité est modélisée comme une ligne distincte de la table *entLogicalTable* de la base ENTITY-MIB. Notez que, contrairement au modèle physique, la logique est plate au lieu d'être hiérarchique.

La base ENTITY-MIB ne distingue pas les différentes classes d'objets logiques, contrairement aux objets physiques. La base SUN-PLATFORM-MIB contient une hiérarchie de classes d'objets logiques, décrite au Chapitre 7.

Les informations de la table *entLogicalTable* peuvent être utilisées pour prendre en charge le multi-scoping à l'aide d'un autre contexte de dénomination. Cependant, cette fonctionnalité n'est pas intégrée à ce produit. Les informations particulièrement importantes sont *entLogicalDescription* et *entLogicalTAddress*, cette dernière fournissant l'adresse IP via laquelle vous pouvez accéder à l'entité logique.

## Mappage d’une hiérarchie logique et physique

La base ENTITY-MIB assure un mappage entre des objets logiques et les objets physiques dont elle est composée. Vous utilisez, pour cela, la table *entLPMappingTable*, table bidimensionnelle (similaire à la table *entPhysicalContainsTable*) qui identifie les entités physiques formant une entité logique donnée. Ces entités physiques sont identifiées par leur index *entLPPhysicalIndex*, équivalent à *entPhysicalIndex*.

Bien que cette table puisse potentiellement représenter toutes les entités physiques associées à une entité logique donnée, en principe, seule l’entité physique contenante est référencée. Par exemple, pour qu’une entité logique soit créée par un module physique, le mappage ne référence que le module, et non toutes les entités physiques qu’il contient.

## Modèle d’événement et d’alarme

La base ENTITY-MIB contient une simple notification SNMP, *entConfigChange*, utilisée pour signaler toutes les modifications apportées à l’une des tables de la MIB. Elle est définie pour fournir un trap maximum toutes les cinq secondes.

La base SUN-PLATFORM-MIB définit des notifications plus spécifiques, décrites au Chapitre 8.

## La base SUN-PLATFORM-MIB

La base SUN-PLATFORM-MIB :

- Étend la table Entité physique pour représenter de nouvelles classes de composants
- Étend la table Entité logique pour représenter une plate-forme et des objets de serveur à valeur élevée

---

**Remarque –** Tous les objets de la base SUN-PLATFORM-MIB portent le préfixe *sunPlat* afin de les distinguer.

---

# Extensions de la table Modèle physique

La base SUN-PLATFORM-MIB contient des attributs supplémentaires de classes qui ne sont pas représentées dans la table Entité physique. Elle étend la table Entité physique en ajoutant les extensions de table suivantes, peuplées de manière éparse :

- Extension de la table Équipement

  Celle-ci permet d'étendre la table Entité physique pour fournir des informations supplémentaires concernant les objets gérés de la classe Équipement. Cette classe est applicable à toutes les Sun Fire B1600 ressources matérielles. Les sous-classes de la classe Équipement sont représentées par d'autres extensions de table.

  - Extension de la table Conteneur d'équipement

    Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés de la classe `container(5)` entPhysicalClass.

  - Extension de la table Pack de circuits

    Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés de la classe `module(9)` entPhysicalClass.

- Extension de la table physique

  Celle-ci permet d'étendre la table Entité physique. Elle permet de compléter la colonne entPhysicalClass de la table Entité physique. Si une ressource contient une classe entPhysicalClass `other(1)`, mais appartient à une classe modélisée par sunPlat, c'est-à-dire une classe Chien de garde ou Périphérique d'alarme, cette table identifie sa classe sunPlatPhysicalClass.

- Extension de la table Capteur

  Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés de la classe entPhysicalClass `sensor(8)`. Les sous-classes de la classe Capteur sont représentées par d'autres extensions de table et sont identifiées par cette table à l'aide de la classe sunPlatSensorClass.

  - Extension de la table Capteur binaire

    Celle-ci permet d'étendre l'extension de la table Capteur. Elle fournit des informations supplémentaires concernant les objets gérés des classes entPhysicalClass `sensor(8)` et sunPlatSensorClass `binary(1)`.

  - Extension de la table Capteur numérique

    Celle-ci permet d'étendre l'extension de la table Capteur. Elle fournit des informations supplémentaires concernant les objets gérés des classes entPhysicalClass `sensor(8)` et sunPlatSensorClass `numeric(2)`.

  - Extension de la table Capteur de valeurs discrètes

    Celle-ci permet d'étendre l'extension de la table Capteur. Elle fournit des informations supplémentaires concernant les objets gérés des classes entPhysicalClass `sensor(8)` et sunPlatSensorClass `discrete(3)`.

- Extension de la table Ventilateur

  Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés de la classe entPhysicalClass `fan(7)`.

- Extension de la table Alarme

  Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés des classes entPhysicalClass `other(1)` et sunPlatPhysicalClass `alarm(8)`.

- Extension de la table Chien de garde

  Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés des classes entPhysicalClass `other(1)` et sunPlatPhysicalClass `watchdog(3)`, généralement représentant des voyants DEL d'indication de service.

- Extension de la table Source d'alimentation

  Celle-ci permet d'étendre l'extension de la table Équipement. Elle fournit des informations supplémentaires concernant les objets gérés de la classe entPhysicalClass `powerSupply(6)`.

La TABLEAU 5-3 montre un exemple d'extensions de la table Entité physique. La table entPhysicalIndex (première colonne de la table) est basée sur l'exemple de hiérarchie de ressources matérielles indiqué dans la FIGURE 5-1

| ENTITY-MIB | | | SUN-PLATFORM-MIB | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| entPhysicalIndex | entPhysicalClass | | | | sunPlatPhysicalClass | | sunPlatFanClass | | sunPlatSensorClass | | | | | sunPlatPowerSupplyClass |
| 1 | châssis | | | | | | | | | | | | | |
| 3 | conteneur | | | | | | | | | | | | | |
| 8 | source d'alimentation | | | | | | | | | | | | | G |
| 12 | source d'alimentation | | | | | | | | | | | | | H |
| 2 | ventilateur | | | | | | A | | | | | | | |
| | ventilateur | | | | | | B | | | | | | | |
| | ventilateur | | | | | | C | | | | | | | |
| 5 | capteur | | | | | | | | D | | | | | |
| 7 | capteur | | | | | | | | E | | | | | |
| 11 | capteur | | | | | | | | F | | | | | |
| 6 | module | | | | | | | | | | | | | |
| 9 | autre | | | | alarme | | | | | | | | | |
| 10 | autre | | | | chien de garde | | | | | | | | | |
| 4 | autre | | | | autre | | | | | | | | | |
| | | | | | | | | | | | | | | |
| entPhysicalTable | | sunPlatEquipmentTable | sunPlatEquipmentHolderTable | sunPlatCircuitPackTable | sunPlatPhysicalTable | sunPlatWatchdogTable | sunPlatFanTable | sunPlatAlarmTable | sunPlatSensorTable | sunPlatBinarySensorTable | sunPlatNumericSensorTable | sunPlatDiscreteSensorTable | sunPlatDiscreteSensorStatusTable | sunPlatPowerSupplyTable |

**TABLEAU 5-3** Extensions de la table Entité physique

**TABLEAU 5-4** Touches pour les extensions de la table Entité physique (TABLEAU 5-3)

| Référence | Description |
|---|---|
| A | Ventilateur |
| B | Réfrigération |
| C | Plaque de refroidissement |
| D | Binaire |
| E | Numérique |
| F | Valeur discrète |
| G | Source d'alimentation |
| H | Batterie |

# Extensions de la table Modèle logique

La base SUN-PLATFORM-MIB contient des attributs supplémentaires de classes qui ne sont pas représentées dans la table Entité logique. Elle étend la table Entité logique en ajoutant les extensions de table suivantes, peuplées de manière éparse :

- Table d'extension de la classe logique

  Elle étend la table entLogicalTable pour définir la classe Entité logique, *SunPlatLogicalClass*, et son état, *sunPlatLogicalStatus*. La table sunPlatLogicalTable est valide pour toutes les entrées de la table entLogicalTable. La sous-classe Système informatique de la classe Logique est représentée par une autre extension de table :

  - Extension de la table Système informatique

    Cette table étend la table entLogicalTable pour contenir des attributs communs aux instances d'un système informatique. La table sunPlatUnitaryComputerSystemTable est valide pour toutes les lignes de la table entLogicalTable contenant une classe sunPlatLogicalClass `computerSystem(2)`.

    Un ensemble d'entrées dans la table Chargement d'informations (*sunPlatInitialLoadInfoTable*) est associé à chaque entité logique du système informatique. Cet ensemble comprend des paramètres permettant de contrôler les paramètres de démarrage du système.

## Tables de journalisation des événements et alarmes

La livraison des traps SNMP n'est pas garantie. Pour cette raison et pour assurer la capacité des applications de gestion à effectuer un suivi précis de l'état actuel des alarmes de la plate-forme, la base MIB établit des listes de problèmes courants pour chaque objet géré. Il s'agit d'une table d'alarmes non résolues pour chaque objet. Elles sont automatiquement supprimées lorsque la condition d'alarme disparaît.

La base SUN-PLATFORM-MIB définit des journaux permettant d'enregistrer des événements ou des alarmes regroupées par type d'événement ou d'alarme, ou par entité affectée. La plate-forme Sun Fire B1600 utilise ces journaux pour établir des listes d'alarmes non résolues pour chaque entité surveillée relative au motif énoncé dans le paragraphe précédent.

Chaque entité de la base MIB pour laquelle les alarmes peuvent être générées (c'est-à-dire toutes les entités physiques et logiques) comporte une entrée dans la table Journal (sunPlatLogTable). Cette table affiche l'état administratif et assure le contrôle des listes de problèmes courants. Les journaux sont activés en permanence et leur taille est illimitée.

La table d'enregistrement des journaux correspondant à chaque entrée de la table Journal peut contenir zéro ou plusieurs entrées. Ces entrées sont décrites en détail au Chapitre 8.

## Enregistrements d'événements

Ces enregistrements font partie des notifications de trap sunPlat. Les modifications du modèle sont communiquées aux applications de gestion à l'aide de deux catégories de traps SNMP ; les événements et les alarmes.

### Événements

- Enregistrement de création d'objets

  Cet enregistrement indique qu'une ressource a été ajoutée au modèle d'objet.
- Enregistrement de suppression d'objets

  Cet enregistrement indique qu'une ressource a été supprimée du modèle d'objet.
- Enregistrement de modification de l'état

  Cet enregistrement indique que l'état de la ressource a changé.

- Enregistrement de la modification de valeur de l'attribut Entier

  Cet enregistrement indique une modification de la caractéristique d'une ressource modélisée par un attribut du type `ENTIER`. L'entier peut être signé ou non signé, selon l'objet affecté.

- Enregistrement de la modification de valeur de l'attribut Chaîne

  Cet enregistrement indique une modification de la caractéristique d'une ressource modélisée par un attribut du type `CHAÎNE D'OCTETS`.

- Enregistrement de la modification de valeur de l'attribut IDO

  Cet enregistrement indique une modification de l'attribut de l'identificateur d'objet du type `IDENTIFICATEUR D'OBJET`.

## Alarmes

- Enregistrement d'alarmes de communication

  Cet enregistrement indique qu'une erreur s'est produite dans les services de communication pris en charge par la ressource.

- Enregistrement d'alarmes d'environnement

  Cet enregistrement indique une condition environnementale concernant la ressource.

- Enregistrement d'alarmes d'équipement

  Cet enregistrement indique qu'une ressource a été endommagée.

- Enregistrement d'alarmes d'erreur de traitement

  Cet enregistrement indique qu'une erreur de traitement ou logicielle est associée à une ressource.

- Enregistrement d'alarmes de qualité de service

  Cet enregistrement indique qu'une alarme de qualité de service s'est déclenchée.

- Enregistrement d'alarmes indéterminées

  Cet enregistrement indique qu'une alarme d'un type inconnu s'est déclenchée.

CHAPITRE 6

# Le modèle physique

Ce chapitre décrit la hiérarchie des classes physiques sunPlat et comment les classes d'objets physiques gérés définies dans le modèle sunPlat sont représentées par la base d'informations SUN-PLATFORM-MIB.

Le chapitre contient les sections suivantes :



## Hiérarchie des classes physiques sunPlat

La FIGURE 6-1 illustre la hiérarchie de l'héritage des classes sunPlat utilisée pour modéliser les ressources matérielles dans la plate-forme Sun Fire B1600.

La super-classe Entité physique contient un attribut permettant de définir la relation entre des objets gérés. Elle contient également des attributs SNMP standard correspondant à des attributs de la classe Équipement.

La classe Équipement sunPlat provient de la super-classe Entité physique pour fournir les attributs supplémentaires définis dans les classes correspondantes qui sont applicables à la surveillance par défaut.

Les classes Conteneur de l'équipement sunPlat et Pack de circuits sunPlat proviennent de la super-classe Équipement sunPlat pour représenter des réceptacles et les composants qui y sont respectivement connectés.

La classe Équipement sunPlat est plus spécialisée pour contenir les classes issues de DMTF.

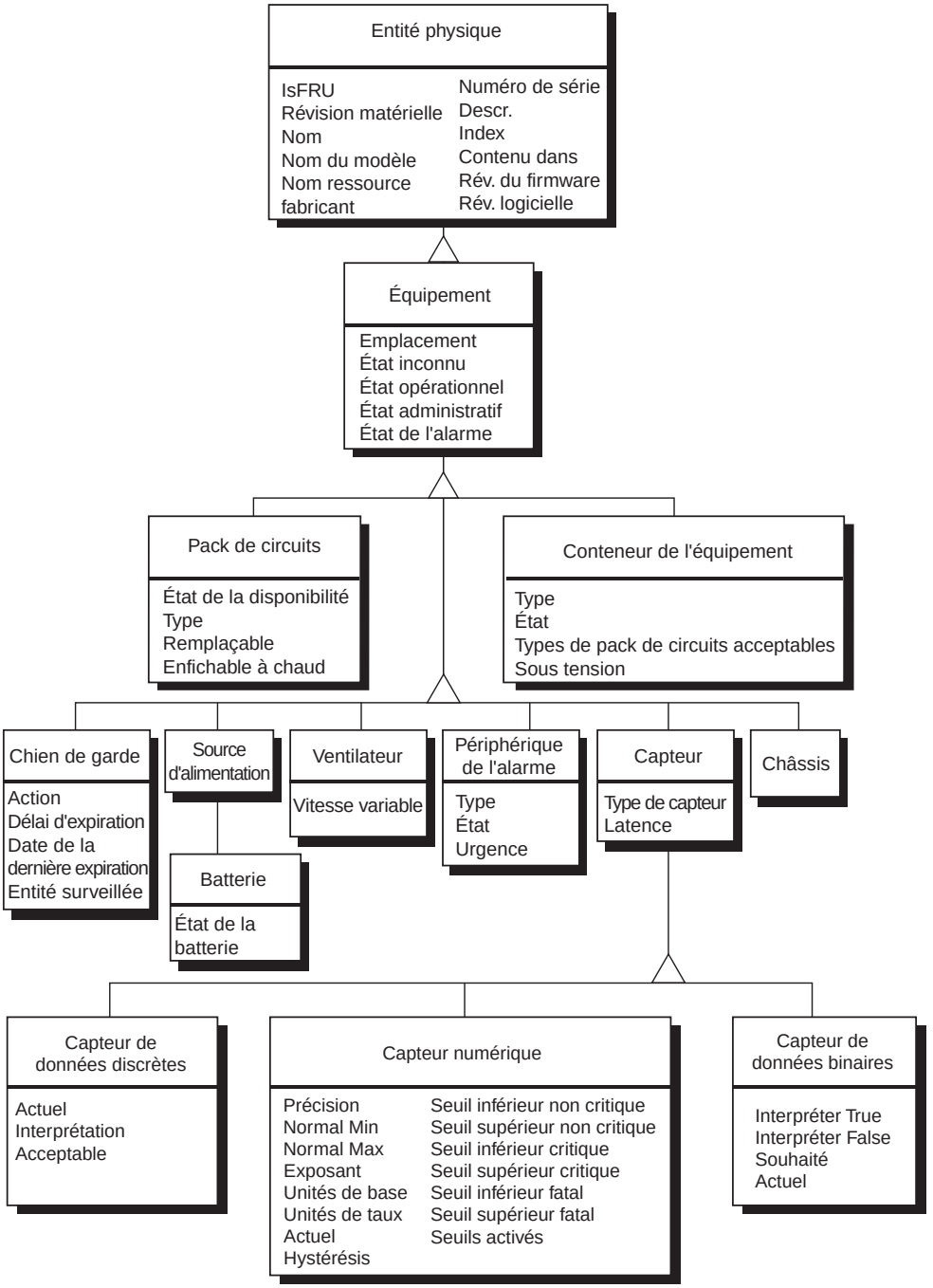


**FIGURE 6-1** Schéma de la classe Héritage de ressources physiques sunPlat

# Définitions des classes sunPlat

Les attributs des classes sunPlat permettent de représenter les caractéristiques des ressources matérielles. La disponibilité et la capacité de fonctionnement de la ressource du gestionnaire sont représentées par l'*état* de l'objet géré. D'autres classes sunPlat possèdent une série d'attributs qui reflètent certains aspects de l'état de l'objet géré.

## Entité physique

La super-classe Entité physique permet de représenter les caractéristiques propres à toutes les ressources.

---

**Remarque –** Le préfixe *entPhysical* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Descr.***
  Il s'agit d'une chaîne textuelle contenant le nom connu de la ressource. Ce nom est celui qui est généralement utilisé pour décrire la ressource dans la documentation du produit, sur des légendes du produit ou éventuellement le nom stocké dans le firmware.
- ***Is FRU***
  Il s'agit d'une valeur booléenne déterminant si la ressource est une unité remplaçable par l'utilisateur. Seules les ressources matérielles de la classe *sunPlatCircuitPack* sont considérées comme des unités remplaçables par l'utilisateur.
- ***Révision matérielle***
  Il s'agit d'une chaîne textuelle contenant les informations sur la révision matérielle de la ressource effectuée par le fabricant. Toutes les ressources matérielles ne sont pas associées à des informations de révision matérielle.
- ***Nom***
  Il s'agit d'une chaîne textuelle contenant le nom logique identifiant la ressource auprès du système d'exploitation et des utilitaires associés. Ce nom peut être un noeud de périphérique ou un nom défini utilisé par des utilitaires du système, le cas échéant. Toutes les ressources ne reçoivent pas un nom de périphérique.
- ***Nom du modèle***
  Il s'agit d'une chaîne textuelle contenant la référence produit ou la description du produit du fabricant visibles du client. Toutes les ressources matérielles ne sont pas associées à des références ou des descriptions produit.
- ***Num. de série***
  Il s'agit d'une chaîne textuelle contenant le numéro de série de la ressource du fabricant. Toutes les ressources matérielles ne sont pas associées à des numéros de série.

- ***Nom ressource fabricant***
  Il s'agit d'une chaîne textuelle contenant le nom de la ressource attribué par le fabricant. Toutes les ressources matérielles ne sont pas associées à un nom de fabricant.

La super-classe Entité physique contient également des attributs permettant de décrire la hiérarchie des ressources matérielles :

- ***Classe***
  Ce type numéroté contient une indication du type matériel général d'une ressource physique particulière. Les valeurs de cette classe prises en charge sont définies par la base ENTITY-MIB. Cet attribut peut être utilisé comme une indication des extensions de tables appropriées pour l'objet géré. Le mappage entre les classes ENTITY-MIB et les classes sunPlat est organisé, comme indiqué dans la TABLEAU 6-1:

**TABLEAU 6-1** Mappage de l'attribut « Classe » de la super-classe Entité physique

| entPhysicalClass | Classe sunPlat |
|---|---|
| chassis(3) | *Châssis sunPlat* |
| backplane(4) | Non implémenté |
| container(5) | *Conteneur de l'équipement sunPlat* |
| powerSupply(6) | *Source d'alimentation sunPlat* |
| fan(7) | *Ventilateur sunPlat* |
| sensor(8) | *Capteur sunPlat*, plus les sous-classes |
| module(9) | *Pack de circuits sunPlat* |
| port(10) | Non implémenté |
| stack(11) | Non implémenté |
| other(1) | *Équipement sunPlat*, plus les sous-classes |
| unknown(2) | Non implémenté |

- ***Index***
  Cet entier identifie par un nom unique l'entrée de la table Entité physique qui identifie l'objet géré. Les valeurs ne sont pas préalablement affectées et peuvent varier à chaque invocation de l'agent.
- ***Contenu dans***
  Cet entier représente l'attribut *Index* de l'objet géré contenant cet objet géré. Par conséquent, l'attribut modélise la relation entre les objets gérés.

---

**Remarque –** L'objet situé à la racine de la hiérarchie de confinement physique (généralement un châssis) ne fait pas physiquement partie d'une autre entité représentée dans la table. Pour cette raison, sa valeur *entPhysicalContainedIn* est définie à `0`.

---

- ***Rév. du firmware***
  Il s'agit d'une chaîne textuelle contenant les informations sur la révision du firmware de la ressource du fabricant. Toutes les ressources matérielles ne sont pas associées à des informations de révision du firmware.
- ***Rév. logicielle***
  Il s'agit d'une chaîne textuelle contenant les informations sur la révision logicielle de la ressource effectuée par le fabricant. Toutes les ressources matérielles ne sont pas associées à des informations de révision logicielle.

# Classe Équipement sunPlat

La classe Équipement sunPlat permet de représenter les caractéristiques propres à toutes les ressources matérielles. Cette classe contient les attributs représentant les informations sur la configuration et l'état de santé général. Elle fait l'objet d'une sous-classe pour fournir des informations de configuration plus détaillées et des données de surveillance pour des types de ressources spécifiques.

La chaîne entPhysicalClass dépend de la sous-classe représentée.

La classe Équipement sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatEquipment* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***État administratif***
  Cet attribut en lecture-écriture adopte une des valeurs numérotées suivantes représentant l'état administratif actuel de la ressource :
  - `locked(1)`
  - `unlocked(2)`
  - `shuttingDown(3)`
- ***État opérationnel***
  Cet attribut en lecture seule est un type numéroté indiquant si la ressource est physiquement installée et capable de fournir des services. L'attribut contribue à l'*état* de l'objet géré et peut adopter les valeurs indiquées dans la TABLEAU 6-2.

**TABLEAU 6-2** Valeurs de l'attribut État opérationnel

| Valeurs de l'attribut | Description |
|---|---|
| `disabled(1)` | La ressource est totalement inopérable et incapable de fournir le service à l'utilisateur. |
| `enabled(2)` | La ressource est partiellement ou totalement opérable et accessible. |

- ***État de l'alarme***
  Cet attribut en lecture seule adopte une valeur numérotée représentant l'état actuel de l'alarme de la ressource. Il indique le niveau de sévérité le plus élevé d'une alarme non résolue sur l'objet géré. Il peut adopter les valeurs suivantes :
  - `critical(1),`
  - `major(2)`
  - `minor(3),`
  - `indeterminate(4),`
  - `warning(5),`
  - `pending(6),`
  - `cleared(7)`
- ***État inconnu***
  Cet attribut en lecture seule indique s'il est possible que les autres attributs de l'état ne reflètent pas l'état réel de la ressource. L'attribut adopte une valeur booléenne déterminant si l'objet géré est capable de signaler de manière concise les erreurs dans la ressource. Si la ressource est véritablement incapable de refléter son *état*, cet attribut est défini comme `réel`.
- ***Nom de l'emplacement***
  Cet attribut en lecture seule contient un identificateur d'emplacement de la ressource. Pour les ressources directement contenues dans le châssis, cet attribut est associé à des légendes sur les emplacements et une documentation produit, ou fournit une indication géographique du positionnement de la ressource au sein du châssis. D'autres ressources matérielles possèdent généralement un *emplacement* correspondant au *nom* de l'objet géré de la ressource dans laquelle il est contenu.

## Classe Pack de circuits sunPlat

La classe Pack de circuits sunPlat permet de représenter les caractéristiques propres à une ressource remplaçable ou à une unité remplaçable par l'utilisateur. Une ressource remplaçable est définie comme un module matériel dont l'objectif est d'emballer des composants matériels internes dans un facteur de forme reconnu. Généralement, une unité remplaçable par l'utilisateur possède un facteur de forme défini et un aspect physique. Il peut s'agir d'une unité amovible enfichable, branchée à un connecteur. Il peut s'agit d'une unité fixe insérée dans une baie. L'unité peut encore être intégrée à un tiroir, un rack ou une étagère.

Cette classe inclut la commande entPhysicalClass `module(9)`.

La classe Pack de circuits sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatCircuitPack* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Type***
  Cet attribut en lecture seule est une chaîne textuelle permettant d'évaluer la compatibilité de la ressource avec son conteneur. Cet attribut peut identifier les fonctionnalités et les caractéristiques du facteur de forme de la ressource.
- ***État de la disponibilité***
  Cet attribut en lecture seule identifie de manière détaillée l'*état opérationnel* de l'objet géré. Il s'agit d'un objet utilisant la syntaxe BITS et peut adopter la valeur zéro ou une autre valeur de l'ensemble de valeurs, indiqué dans la TABLEAU 6-3. Toutes ces valeurs ne sont pas applicables à chaque classe d'objet géré. Cet attribut contribue à l'*état* de l'objet géré.

**TABLEAU 6-3** Valeurs de l'attribut État de la disponibilité

| Valeurs de l'attribut | Nbre de bits | Hex. | Description |
|---|---|---|---|
| `inTest(0)` | `0` | `80` | La ressource est soumise à un test. |
| `failed(1)` | `1` | `40` | La ressource fait l'objet d'une panne interne l'empêchant de fonctionner. L'*état opérationnel* est `disabled(1)`. |
| `powerOff(2)` | `2` | `20` | La ressource doit être sous tension pour être appliquée et n'est pas actuellement sous tension. |
| `offLine(3)` | `3` | `10` | La ressource requiert l'exécution d'une opération de routine pour être placée en ligne et la rendre accessible. L'*état opérationnel* est `disabled(1)`. |
| `offDuty(4)` | `4` | `08` | La ressource a été désactivée par un processus de contrôle interne. |
| `dependency(5)` | `5` | `04` | La ressource ne peut pas fonctionner car une autre ressource dont elle dépend n'est pas accessible. L'*état opérationnel* est `disabled(1)`. |
| `degraded(6)` | `6` | `02` | Le service accessible dans la ressource est d'une certaine manière défaillant. (ex. : vitesse ou capacité de fonctionnement altérée). Toutefois, le service reste disponible dans la ressource. L'*état opérationnel* est `enabled(2)`. |
| `notInstalled(7)` | `7` | `01` | La ressource représentée par l'objet géré n'est pas présente ou est incomplète. L'*état opérationnel* est `disabled(1)`. |

- ***Remplaçable***
  Cet attribut en lecture seule adopte une valeur booléenne indiquant si la ressource est une unité remplaçable.
- ***Enfichable à chaud***
  Cet attribut en lecture seule adopte une valeur booléenne indiquant si la ressource remplaçable est enfichable à chaud.

# Conteneur de l'équipement sunPlat

Laclasse Conteneur de l'équipement sunPlat permet de représenter les caractéristiques des ressources matérielles capables de conserver les ressources matérielles amovibles.

Cette classe inclut la commande entPhysicalClass `container(5)`.

La classe Conteneur de l'équipement sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatEquipmentHolder* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Type***
  Cet attribut en lecture seule est une valeur numérotée représentant le type de conteneur de la ressource, comme indiqué dans la TABLEAU 6-4:

**TABLEAU 6-4** Valeurs de l'attribut Conteneur de l'équipement

| Valeurs de l'attribut | Description |
|---|---|
| `bay(1)` | Une baie est généralement une unité d'espace vertical située dans un rack contenant des étagères ou des tiroirs permettant de conserver l'équipement de télécommunications. sunPlat interprète son utilisation dans un châssis comme un réceptacle physique exigeant des câbles pour les connexions de signaux |
| `shelf(2)` | Support horizontal ou sous-rack destiné à soutenir l'équipement de télécommunications dans un rack. |
| `drawer(3)` | Enceinte horizontale destinée à soutenir l'équipement de télécommunications dans un rack. |
| `slot(4)` | Réceptacle physique doté d'un connecteur intégral pour des connexions de signaux de l'équipement amovible. |
| `rack(5)` | Un rack est l'infrastructure de soutien chargée de soutenir l'équipement de télécommunications, les conteneurs et les systèmes de gestion des câbles au sein d'une enceinte autonome. |

- ***Types acceptables***
  Cet attribut en lecture seule est une liste de chaînes textuelles représentant les types de ressources amovibles (pack de circuits) pris en charge par le conteneur. Ces types sont testés pour vérifier leur compatibilité avec l'attribut *Type* de la ressource amovible.
- ***État***
  Cet attribut en lecture seule est un type numéroté indiquant l'état du conteneur relativement aux ressources matérielles remplaçables (packs de circuits) qu'il peut contenir, comme indiqué dans la TABLEAU 6-5.

**TABLEAU 6-5** Valeurs de l'attribut État du conteneur de l'équipement

| Valeurs de l'attribut | Description |
|---|---|
| `holderEmpty(1)` | Aucune ressource amovible n'est présente dans le conteneur |
| `inTheAcceptableList(2)` | Le conteneur inclut une ressource amovible faisant partie de la liste des *types de packs de circuits acceptables* |
| `notInTheAcceptableList(3)` | Le conteneur inclut une ressource amovible reconnaissable par l'élément réseau mais ne faisant pas partie de la liste des *types de packs de circuits acceptables* |
| `unknownType(4)` | Le conteneur inclut une ressource amovible non identifiable |

- ***Sous tension***
  Cet attribut en lecture-écriture est un type numéroté indiquant l'état de l'alimentation de la ressource. Les valeurs possibles sont :
  - `other(1)`
  - `unknown(2)`
  - `powerOff(3)`
  - `powerOn(4)`

# Source d'alimentation sunPlat

La classe Source d'alimentation sunPlat permet de représenter une source d'alimentation. Elle ne détaille pas les caractéristiques de la classe Équipement sunPlat. Une source d'alimentation contient généralement des capteurs représentant les propriétés surveillés, comme par exemple, les tensions, le courant et la température. Elle peut également contenir d'autres ressources matérielles, telles que les ventilateurs. Cette classe est modélisée par des relations entre les objets gérés.

Si une source d'alimentation est une ressource amovible, elle est modélisée dans un objet géré de la classe Pack de circuits sunPlat.

Cette classe inclut la commande entPhysicalClass `powerSupply(6)`.

La classe Source d'alimentation sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatPowerSupply* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Classe***

  Cet attribut en lecture seule est un type numéroté indiquant la classe de la source d'alimentation et adopte les valeurs suivantes :

  - `other(1)`
  - `powerSupply(2)`
  - `battery(3)`

## Batterie sunPlat

La classe Batterie sunPlat permet de représenter une source d'alimentation fournissant l'énergie électrique à partir d'une batterie.

Cette classe inclut les classes entPhysicalClass `powerSupply(6)` et SunPlatPowerSupplyClass `battery(3)`.

La classe Batterie sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatBattery* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***État***

  Cet attribut en lecture seule est un type numéroté indiquant l'état de la batterie et adopte les valeurs suivantes :

  - `other(1)`
  - `unknown(2)`
  - `fullyCharged(3)`
  - `low(4)`
  - `critical(5)`
  - `charging(6)`
  - `chargingAndHigh(7)`
  - `chargingAndLow(8)`
  - `chargingAndCritical(9)`
  - `undefined(10)`
  - `partiallyCharged(11)`

# Chien de garde sunPlat

La classe Chien de garde sunPlat permet de représenter les caractéristiques des ressources matérielles du minuteur permettant au matériel de surveiller l'état du système d'exploitation ou des applications.

Cette classe inclut les classes entPhysicalClass `other(1)` et SunPlatPhysicalClass `watchdog(3)`.

La classe Chien de garde sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatWatchdog* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Délai d'expiration***
  Cet attribut en lecture seule est un entier indiquant l'intervalle en millisecondes au-delà duquel le chien de garde expire s'il n'est pas réinitialisé.
- ***Action***
  Cet attribut en lecture seule est un type numéroté représentant l'action menée par le chien de garde s'il n'est pas réinitialisé au-delà de la période spécifiée dans le *délai d'expiration*. Les valeurs possibles sont indiquées dans la TABLEAU 6-6.

**TABLEAU 6-6** Valeurs de l'attribut Action du chien de garde

| Action | Description |
|---|---|
| `statusOnly(1)` | Le chien de garde est lisible par le logiciel mais n'exécute aucune action |
| `systemInterrupt(2)` | Le chien de garde génère une interruption matérielle du système en cours de surveillance |
| `systemReset(3)` | Le chien de garde réinitialise le système surveillé |
| `systemPowerOff(4)` | Le chien de garde met le système en cours de surveillance hors tension |
| `systemPowerCycle(5)` | Le chien de garde met le système en cours de surveillance hors tension, puis de nouveau sous tension |

- ***Date de la dernière expiration***
  Cet attribut en lecture seule indique la date et l'heure à laquelle le chien de garde a expiré pour la dernière fois.
- ***Entité surveillée***
  Cet attribut en lecture seule est un type numéroté représentant les entités pouvant être surveillées par le chien de garde. Les valeurs possibles sont :
  - `unknown(1)`
  - `other(2)`

- operatingSystem(3)
- operatingSystemBootProcess(4)
- operatingSystemShutdownProcess(5)
- firmwareBootProcess(6)
- biosBootProcess(7)
- application(8)
- serviceProcessors(9)

## Alarme sunPlat

La classe Alarme sunPlat permet de représenter les caractéristiques des ressources matérielles qui émettent des indications relatives à des problèmes, comme par exemple, des alertes sonores, des voyants DEL, des relais, des vibrateurs et des alarmes logicielles.

Cette classe inclut les commandes entPhysicalClass `other(1)` et SunPlatPhysicalClass `alarm(2)`.

La classe Alarme sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatAlarm* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Type***
  Cet attribut en lecture seule est un type numéroté représentant les moyens par lesquels la condition Alarme est communiquée. Les valeurs possibles sont indiquées dans la TABLEAU 6-7.

**TABLEAU 6-7** Valeurs de l'attribut Type d'alarme

| Valeurs de l'attribut | Description |
|---|---|
| `other(1)` | Le type de périphérique d'alarme n'est pas l'un des types suivants |
| `audible(2)` | L'alarme cause une modification audible sur le périphérique |
| `visible(3)` | L'alarme cause une modification visible sur le périphérique |
| `motion(4)` | L'alarme cause un mouvement sur le périphérique |
| `switch(5)` | L'alarme cause une modification du signal électrique |

- ***État***
  Cet attribut en lecture-écriture est un type numéroté représentant l'état de l'alarme. Les valeurs possibles sont indiquées dans la TABLEAU 6-8.

**TABLEAU 6-8** Valeurs de l'attribut État de l'alarme

| Valeurs de l'attribut | Description |
|---|---|
| `unknown(1)` | L'état de l'alarme est indéfini ou inobservable |
| `off(2)` | L'alarme est inactive |
| `steady(3)` | L'alarme est active |
| `alternating(4)` | L'alarme bascule entre les états inactif et actif |

- ***Urgence***
  Cet attribut en lecture-écriture est un type numéroté indiquant la fréquence relative avec laquelle l'alarme s'allume, vibre et/ou émet des sons. Les valeurs possibles sont :
  - `other(1)`
  - `unknown(2)`
  - `notSupported(3)`
  - `informational(4)`
  - `nonCritical(5)`
  - `critical(6)`
  - `unrecoverable(7)`

# Ventilateur sunPlat

La classe Ventilateur sunPlat permet de représenter les caractéristiques des périphériques de refroidissement actifs. Un ventilateur contient généralement un capteur représentant la vitesse de rotation. Ce dernier st modélisé à l'aide d'une relation de confinement physique entre l'objet géré Ventilateur sunPlat et un objet géré par un tachymètre appartenant à la classe Capteur sunPlat.

Cette classe inclut la commande entPhysicalClass `fan(7)`.

La classe Ventilateur sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatFan* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Classe***
  Cet attribut en lecture seule est un type numéroté indiquant la classe du périphérique de refroidissement et adopte les valeurs suivantes :
  - `other(1)`
  - `fan(2)`
  - `refrigeration(3)`
  - `heatPipe(4)`

# Capteur sunPlat

La super-classe Capteur sunPlat permet de représenter les caractéristiques propres aux ressources matérielles mesurant les propriétés d'autres ressources matérielles.

Cette classe inclut la commande entPhysicalClass `sensor(8)`.

La classe Capteur sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatSensor* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Classe***
  Cet attribut en lecture seule est un type numéroté indiquant la classe du capteur et adopte les valeurs suivantes :
  - `binary(1)`
  - `numeric(2)`
  - `discrete(3)`
- ***Type***
  Cet attribut en lecture seule est un type numéroté identifiant la propriété mesurée par le capteur. Quelques unes des valeurs possibles de la classe *Type* sont indiquées dans la TABLEAU 6-9.

**TABLEAU 6-9** Valeurs de l'attribut Type de capteur

| Type | Description |
|---|---|
| `temperature(3)` | Capteur permettant de mesurer la température de l'environnement |
| `voltage(4)` | Capteur permettant de mesurer la tension électrique |
| `current(5)` | Capteur permettant de mesurer le courant électrique |
| `tachometer(6)` | Capteur permettant de mesurer la vitesse ou les révolutions d'un périphérique |
| `counter(7)` | Capteur d'utilité générale permettant de comptabiliser des événements définis |

- ***Latence***
  Cet attribut en lecture seule indique les informations suivantes :
  - le lieu d'interrogation du capteur ; cet entier représente l'intervalle de mise à jour exprimé en millisecondes.
  - l'orientation du capteur par l'événement ; cette valeur représente la latence maximale attendue dans le traitement de cet événement.

# Capteur binaire sunPlat

Une classe Capteur binaire sunPlat permet de représenter les caractéristiques des capteurs qui renvoient une sortie binaire. Elle augmente la table sunPlatSensor pour fournir les attributs spécifiques aux capteurs binaires.

Cette classe inclut les commandes entPhysicalClass `sensor(8)` et SunPlatSensorClass `binary(1)`.

La classe Capteur binaire sunPlat comporte les attributs suivants :

---

**Remarque –** le préfixe *sunPlatBinarySensor* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Actuel***
  Cet attribut en lecture seule adopte une valeur booléenne indiquant la valeur du capteur la plus récente.
- ***Attendu***
  Cet attribut en lecture seule adopte une valeur booléenne indiquant la valeur anticipée du capteur.
- ***Interpréter la valeur True***
  Cet attribut en lecture seule est une chaîne textuelle indiquant l'interprétation d'une valeur `true` par le capteur.
- ***Interpréter la valeur False***
  Cet attribut en lecture seule est une chaîne textuelle indiquant l'interprétation d'une valeur `false` par le capteur.

# Capteur numérique sunPlat

Une classe Capteur numérique sunPlat permet de représenter les caractéristiques des capteurs qui renvoient des lectures numériques. Les valeurs du capteur numérique sont approuvées par une unité de mesure, comme défini ci-dessous :

Unité de mesure = *unité de base* * $10^{Exposant}$

Cette désignation autorise des unités de mesure, telles que les milli-ampères et les micro-volts. Si une *unité de taux* est définie, l'unité de mesure est affinée comme ci-dessous :

Unité de mesure = *unité de base* * $10^{Exposant}$ par *unité de taux*

Cette désignation autorise des unités de mesure, telles que les tr / min. et les km / h.

Cette classe inclut les commandes entPhysicalClass `sensor(8)` et SunPlatSensorClass `numeric`(2).

La classe Capteur numérique sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatNumericSensor* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Unités de base***
  Cet attribut en lecture seule est un type numéroté indiquant l'unité de mesure, avant la validation définie ci-dessus. Exemples de valeurs de ce type :
  - `degC(3)`
  - `volts(6)`
  - `amps(7)`
- ***Exposant***
  Cet attribut en lecture seule est un entier qui multipliait l'*unité de base* par 10. Par exemple, si *sunPlatNumericSensorBaseUnits* est définie en `volts` et *sunPlatNumericSensorExponent* est définie à `-6`, les unités des valeurs renvoyées sont des `micro-volts`.
- ***Unités de taux***
  Cet attribut en lecture seule est un type numéroté indiquant si le capteur mesure une valeur absolue (lorsque la valeur est `none`) ou un taux. Dans ce cas, l'unité spécifiée dans *sunPlatNumericSensorBaseUnits* est exprimée en unité de temps. Par exemple, si *sunPlatNumericSensorBaseUnits* est définie sur `degC` et *sunPlatNumericSensorRateUnits*, sur `perSecond`, la valeur représentée inclut les unités `degC/second`.

  Exemples de valeurs de ce type :
  - `perMicrosecond(2)`
  - `perMillisecond(3)`
  - `perSecond(4)`

- `perMinute(5)`
- `perHour(6)`
- `none(1)`

- ***Actuel***
  Cet attribut en lecture seule est un entier indiquant la valeur du capteur la plus récente.
- ***Normal Min***
  Cet attribut en lecture seule est un entier indiquant le seuil défini en dessous duquel le capteur ne doit pas lire. Cette valeur est exprimée en unités de mesure définies ci-dessus. Cet attribut peut ne pas s'appliquer à certains capteurs.
- ***Normal Max***
  Cet attribut en lecture seule est un entier indiquant le seuil défini au-delà duquel la lecture du capteur ne doit pas se situer. Cette valeur est exprimée en unités de mesure définies ci-dessus. Cet attribut peut ne pas s'appliquer à certains capteurs.
- ***Précision***
  Cet attribut en lecture seule est un entier indiquant le degré d'erreur du capteur pour la propriété mesurée comme un pourcentage à deux décimales. La valeur peut varier selon que la lecture du capteur est linéaire sur sa plage dynamique.
- ***Seuil inférieur non critique***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus bas au niveau duquel une condition `non critique` se produit.
- ***Seuil supérieur non critique***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus haut au niveau duquel une condition `non critique` se produit.
- ***Seuil inférieur critique***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus bas au niveau duquel une condition `critique` se produit.
- ***Seuil supérieur critique***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus haut au niveau duquel une condition `critique` se produit.
- ***Seuil inférieur fatal***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus bas au niveau duquel une condition `fatale` se produit.
- ***Seuil supérieur fatal***
  Cet attribut en lecture seule est un entier indiquant le seuil le plus haut au niveau duquel une condition `fatale` se produit.
- ***Hystérésis***
  Cet attribut en lecture seule décrit l'hystérésis autour des valeurs du seuil.
- ***Seuils activés***
  Cet attribut en lecture seule restaure les paramètres par défaut des capteurs lorsque vous écrivez les données dessus.

# Capteur de valeurs discrètes sunPlat

La classe Capteur de valeurs discrètes sunPlat est utilisée pour les capteurs qui ne peuvent pas être représentés par les classes Capteur numérique sunPlat ou Capteur binaire sunPlat.

Cette classe inclut les commandes entPhysicalClass `sensor(8)` et SunPlatSensorClass `discrete`(3).

La classe comprend deux tables. La table sunPlatDiscreteSensor contient un attribut, *sunPlatDiscreteSensorCurrent*, qui indique l'état actuel du capteur considéré comme un index dans la table sunPlatDiscreteSensorStates.

La classe Capteur discret sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatDiscreteSensorState* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Index***
  Cet attribut en lecture seule sélectionne un numéro qui représente l'index d'une ligne de la table sunPlatDiscreteSensorStates table identifiant l'état de ce capteur.
- ***Interprétation***
  Cet attribut en lecture seule est une chaîne décrivant l'état représenté par la ligne correspondante de la table sunPlatDiscreteSensorStatesTable.
- ***Acceptable***
  Cet attribut en lecture seule adopte une valeur booléenne indiquant si l'état représenté par cette ligne de la table est considéré comme acceptable.

# Châssis sunPlat

La classe Châssis sunPlat permet de représenter le boîtier principal. Elle ne détaille pas les caractéristiques de la classe Équipement sunPlat. Le châssis contient toutes les ressources matérielles modélisées et ne fait partie intégrante d'aucune autre ressource.

Cette classe inclut l'élément entPhysicalClass `chassis(3)`.

CHAPITRE 7

# Le modèle logique

Ce chapitre décrit la hiérarchie des classes logiques sunPlat et comment les classes d'objets gérés définies dans le modèle sunPlat sont représentées par la base SUN-PLATFORM-MIB.

Il contient les sections suivantes :



## Hiérarchie des classes logiques sunPlat

La FIGURE 7-1 indique la hiérarchie de l'héritage des classes logiques sunPlat.

La classe Entité logique contient des informations communes à tous les objets logiques.

La classe Système informatique unitaire ajoute des propriétés relatives à la création de rapports sur l'état de l'alimentation des systèmes informatiques modélisés (par exemple, une lame Sun Fire B100s dans un châssis Sun Fire B1600), qui peut être également utilisée pour forcer la réinitialisation.

La classe Domaine administratif n'ajoute aucune propriété supplémentaire, mais est utilisée pour représenter l'objet logique désignant le contact administratif avec le système modélisé. Dans le cas de la plate-forme Sun Fire B1600, on l'utilise pour représenter le contrôleur du système.

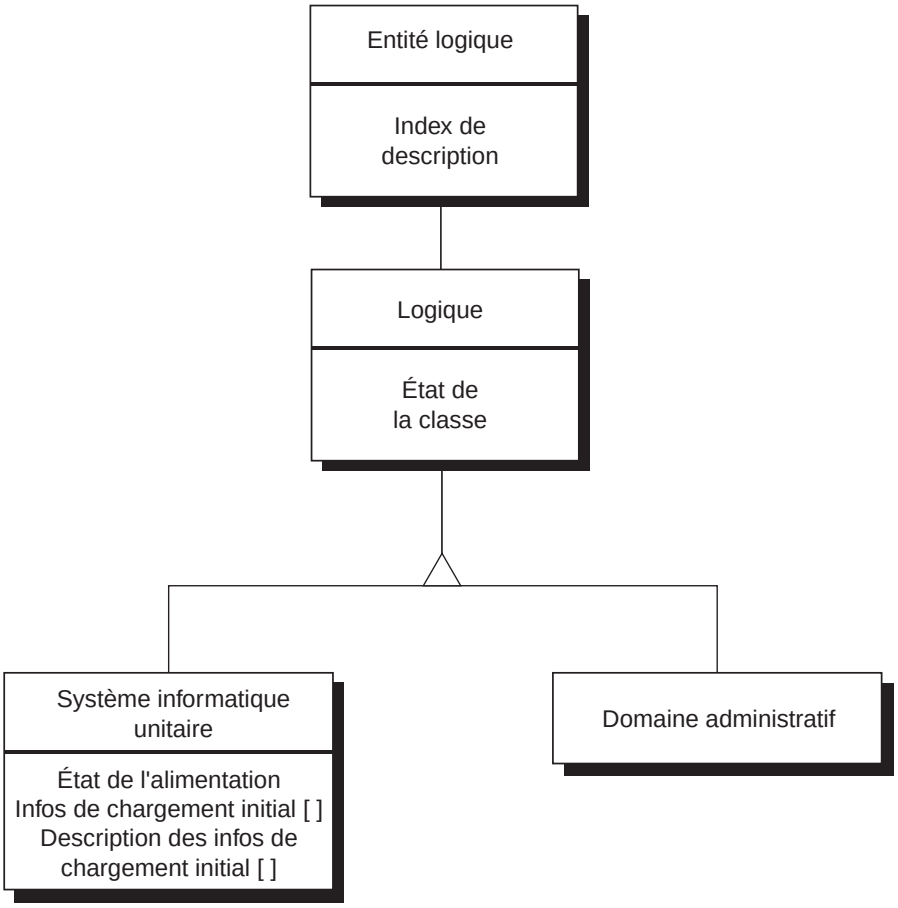


**FIGURE 7-1** Le schéma de la classe d'héritage des ressources logiques sunPlat

# Définitions des classes logiques sunPlat

Les attributs des classes logiques sunPlat permettent de représenter les caractéristiques des ressources logiques. Ces ressources représentent les objets *à valeur élevée*, tels que les domaines d'un système à domaines multiples. La disponibilité et la capacité de fonctionnement de la ressource du gestionnaire sont représentées par l'état de l'objet géré. D'autres classes sunPlat possèdent une série d'attributs qui reflètent certains aspects de l'état des objets gérés.

# Entité logique

Cette classe représente l'entité logique fournissant des informations sur l'identité. Les objets significatifs sont :

---

**Remarque –** Le préfixe *entLogical* a été omis des noms d'objets suivants pour une plus grande clarté.

---

- ***Description***

  Cet objet identifie le type d'objet géré.

- ***TAddress***

  Il contient l'adresse IP et le numéro de port UDP par le biais duquel l'entité peut être directement gérée. Pour une lame Sun Fire B100s d'un système Sun Fire B1600, il contient l'adresse IP de la lame, et le port 161, via lequel l'agent SNMP Solaris standard de la lame est contacté.

# logique

Cette classe représente le type d'état de la ressource représentée par cette entité logique. La classe contient les objets suivants.

---

**Remarque –** Le préfixe *entLogical* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Classe***

  Cet attribut est un paramètre numéroté indiquant le type de classe logique et adopte les valeurs suivantes :

  - `other(1)`
  - `computerSystem(2)`
  - `adminDomain(3)`

- ***État***

  Cet attribut est un paramètre numéroté indiquant l'état de la classe logique. Il peut adopter les valeurs suivantes :

  - `ok(1)`
  - `error(2)`
  - `degraded(3)`
  - `unknown(4)`
  - `predFail(5)`

- `starting(6)`
- `stopping(7)`
- `service(8)`
- `stressed(9)`
- `nonRecover(10)`
- `noContact(11)`
- `lostComm(12)`
- `stopping(13)`

## Système informatique unitaire sunPlat

Les propriétés spécifiques à cette classe sont représentées à l'aide de la table sunPlatUnitaryComputerSystemTable. Elle comporte la classe logique sunPlat nommée `computerSystem(2)`

La classe contient les objets suivants :

---

**Remarque –** Le préfixe *sunPlatUnitaryComputerSystem* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***État de l'alimentation***

  Cet attribut indique l'état actuel de l'alimentation lors de sa lecture. Il assure également le contrôle distant de l'état de l'alimentation, par exemple, pour mettre une lame sous ou hors tension, ou pour synchroniser la mise sous/hors tension pour forcer une réinitialisation.

  Il peut adopter les valeurs suivantes :

  - `unknown(1)`
  - `fullPower(2)`
  - `psLowPower(3)`
  - `psStandby(4)`
  - `psOther(5)`
  - `powerCycle(6)`
  - `powerOff(7)`
  - `psWarning(8)`
  - `hibernate(9)`
  - `softOff(10)`
  - `reset(11)`

- ***Appliquer les paramètres***

  L'écriture dans cette propriété permet d'appliquer un ensemble de paramètres d'amorçage par défaut ou personnalisé.

  Associé à chaque entrée de la table sunPlatUnitaryComputerSystemTable, un ensemble d'entrées de la table *sunPlatInitialLoadInfoTable* définit le paramètre d'amorçage courant et un ensemble alternatif pouvant être appliqué en écrivant dans l'objet *sunPlatUnitaryComputerSystemApplySettings*.

# Domaine administratif sunPlat

Cette classe n'ajoute aucune propriété à la classe Entité logique et n'est donc associée à aucun objet MIB. Elle inclut la classe logique sunPlat appelée `adminDomain(3)`

CHAPITRE 8

# Les notifications sunPlat

Ce chapitre décrit les classes et les attributs de notifications SunPlat, définis dans la base d'informations SUN-PLATFORM-MIB.

Les classes de notifications sunPlat sont des messages asynchrones envoyés par l'agent à des gestionnaires réseau enregistrés. Elles permettent de transmettre les informations relatives aux événements de manière plus efficace en interrogeant les objets gérés.

Le chapitre contient les sections suivantes :



## Hiérarchie de classes de notifications sunPlat

La FIGURE 8-1 indique la hiérarchie de l'héritage des classes de notifications sunPlat.

L'ensemble des classes de notifications est représenté sous forme d'arborescence de classes abstraites et concrètes en utilisant les attributs courants dans ces classes.

# Classes d’enregistrement d’événements sunPlat

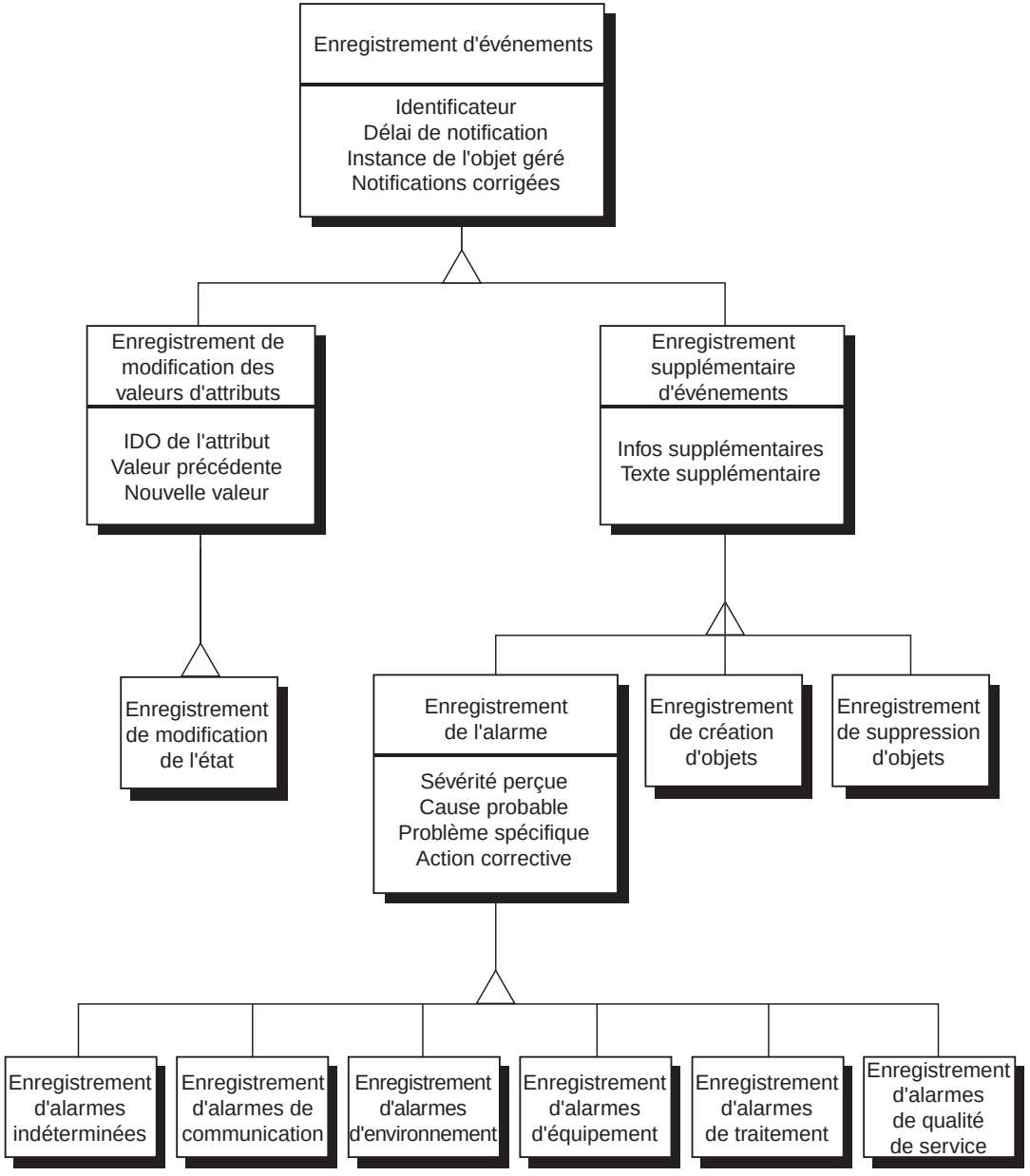


**FIGURE 8-1** Schéma de la classe d’héritage d’enregistrements d’événements

# Définitions des classes sunPlat

## Enregistrement d'événements sunPlat

La super-classe Enregistrement d'événements sunPlat représente les attributs communs à toutes les notifications. Cette classe fait l'objet d'une sous-classe pour fournir des informations supplémentaires relatives à l'événement particulier qu'elle enregistre.

La super-classe Enregistrement d'événements de sunPlat comporte les attributs suivants :

---

**Remarque –** le préfixe *sunPlatLogRecord* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***ID***
  Il s'agit d'un entier permettant d'identifier de manière exclusive la notification et d'une indication de l'ordre dans lequel les notifications ont été générées par l'agent. Notez que l'agent ne garantit pas que la séquence utilisée reflète l'ordre des événements sous-jacents dans lequel les notifications sont générées.
- ***Heure de notification***
  Cette attribut en lecture seule est un horodateur qui identifie actuellement l'heure à laquelle la notification est générée.
- ***Instance d'objet géré***
  Cet attribut en lecture seule est un IDO qui contient une référence directe à une entrée de la base MIB représentant la ressource à laquelle est associée l'événement.
- ***Notifications corrélées***
  Cet attribut en lecture seule est une liste de valeurs d'identificateurs séparés par des virgules, permettant d'identifier les autres événements auxquels cet événement est associé.

## Enregistrement supplémentaire d'événements sunPlat

La super-classe Enregistrement supplémentaire d'événements sunPlat représente les attributs communs aux notifications générées lorsque les événements suivants se produisent :

- Création d'objets
- Suppression d'objets
- Alarmes

Cette classe fait l'objet d'une sous-classe pour fournir des informations supplémentaires applicables à l'événement particulier qu'elle enregistre.

La super-classe Enregistrement supplémentaire d'événements sunPlat comporte les attributs suivants :

---

**Remarque –** le préfixe *sunPlatLogRecord* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***Informations supplémentaires***
  Cet attribut en lecture seule est un IDO d'objet facultatif pouvant fournir des informations supplémentaires relatives à cette notification.
- ***Texte supplémentaire***
  Chaîne textuelle facultative d'un attribut en lecture seule fournissant des informations supplémentaires sur cette notification et permettant d'identifier le composant affecté par son label et le nom `entPhysical`.

## Enregistrement de création d'objets sunPlat

La classe Enregistrement de création d'objets sunPlat permet d'indiquer l'ajout d'une ressource à la hiérarchie en dessous de la ressource associée ; ceci peut être du à un événement d'enfichage à chaud. Les attributs *Informations supplémentaires* contiennent l'IDO de l'entrée *Table Entité physique* représentant la ressource ajoutée.

Les objets logiques sont créés sous une instance d'objet géré `0.0`.

## Enregistrement de suppression d'objets sunPlat

La sous-classe Enregistrement de suppression d'objets sunPlat permet d'indiquer la suppression d'une ressource de la hiérarchie en dessous de la ressource associée. Les attributs *Informations supplémentaires* contiennent l'IDO de l'entrée *Table Entité physique* représentant la ressource supprimée.

---

**Remarque –** Cet IDO n'est plus valide mais peut être toujours utilisé par le gestionnaire qui le reçoit

---

# Enregistrement d’alarmes sunPlat

La super-classe Enregistrement d’alarmes sunPlat représente les attributs communs à toutes les notifications représentant des alarmes.

Cette classe est sous-divisée pour une identification de la classe d’alarme qui a été déclenchée.

La super-classe Enregistrement d’alarmes sunPlat comporte les attributs suivants :

---

**Remarque –** le préfixe *sunPlatAlarmRecord* a été omis des noms d’attributs suivants pour une plus grande clarté.

---

- ***Niveau de sévérité perçu***
  Cet attribut en lecture seule est un type numéroté définissant six niveaux de sévérité qui indiquent à quel niveau le service de la ressource a été affecté par le problème. Ces valeurs sont indiquées dans la TABLEAU 8-1.

**TABLEAU 8-1** Valeurs des niveaux de sévérité d’enregistrement d’alarmes sunPlat perçus

| Niveau de sévérité perçu | Description |
|---|---|
| `indeterminate(1)` | Le niveau de sévérité de l’alarme ne peut pas être déterminé. |
| `critical(2)` | Une condition affectant un service s’est produite et une action corrective immédiate est requise. |
| `major(3)` | Une condition affectant un service s’est produite et une action corrective urgente est requise. |
| `minor(4)` | Une condition n’affectant pas un service s’est produite et une action corrective doit être effectuée afin d’empêcher l’émergence d’une condition plus grave. |
| `warning(5)` | Une condition d’erreur affectant ou empêchant potentiellement un service a été détectée et une action doit être effectuée afin d’empêcher l’émergence d’une condition plus sérieuse. |
| `cleared(6)` | Cette condition efface toutes les alarmes liées à cette ressource de la même classe d’alarmes partageant une *cause probable* et un *problème spécifique* communs (le cas échéant). |

- ***Cause probable***
  Cet attribut en lecture seule est un type numéroté facultatif fournissant des informations détaillées sur le type de condition qui a causé la génération de l’alarme. Exemples de valeurs de ce type :
  - `coolingSystemFailure(134)`
  - `IODeviceError(75)`
  - `powerProblem(58)`
  - `softwareProgramError(283)`

- ***Problème spécifique***
  Cet attribut en lecture seule est une chaîne textuelle facultative identifiant les détails de la *cause probable* de l'alarme.
- ***Action corrective***
  Cet attribut en lecture seule est une chaîne qui répertorie les actions correctives recommandées.

## Enregistrement d'alarmes indéterminées sunPlat

La classe Enregistrement d'alarmes indéterminées sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet d'enregistrer les alarmes qui n'appartiennent pas aux classes suivantes :

## Enregistrement d'alarmes de communication sunPlat

La classe Enregistrement d'alarmes de communication sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet de signaler que la ressource associée a détecté une erreur de communication.

## Enregistrement d'alarmes d'environnement sunPlat

La classe Enregistrement d'alarmes d'environnement sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet de signaler que la ressource associée a détecté un problème dans l'environnement.

## Enregistrement d'alarmes d'équipement sunPlat

La classe Enregistrement d'alarmes d'équipement sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet de signaler que la ressource associée a détecté une panne.

## Enregistrement d'alarmes de traitement sunPlat

La classe Enregistrement d'alarmes de traitement sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet de signaler que la ressource associée a détecté uneerreur de logiciel ou de traitement.

## Enregistrement d'alarmes de qualité de service sunPlat

La classe Enregistrement d'alarmes de qualité de service sunPlat ne détaille pas les informations fournies par la classe Enregistrement d'alarmes sunPlat. Cette classe permet de signaler que la ressource associée a détecté un changement dans la qualité du service.

# Enregistrement de modification des valeurs d'attributs sunPlat

La super-classe Enregistrement de modification des valeurs d'attributs sunPlat désigne les attributs supplémentaires communs aux notifications représentant des changements d'attributs dans la ressource associée.

Cette classe fait l'objet d'une sous-classe pour chacun des types d'attributs possibles.

La super-classe Enregistrement de modification des valeurs d'attributs sunPlat comporte les attributs suivants :

---

**Remarque –** Le préfixe *sunPlatLogRecordChange* a été omis des noms d'attributs suivants pour une plus grande clarté.

---

- ***IDO***
  Cet attribut en lecture seule est un IDO qui fait une référence directe à un objet dans la table Entité physique ou la table Entité logique représentant l'attribut de l'objet géré dont la valeur a changé.

En fonction de la syntaxe de l'attribut affecté, les valeurs anciennes et nouvelles sont représentées à l'aide de l'une des paires d'objets suivante :

- ***Nouvel entier***
  Cet attribut en lecture seule identifie la nouvelle valeur de l'`ENTIER` de l'attribut modifié de l'objet géré. Le type signé ou non signé correspond à celui de l'attribut modifié.
- ***Ancien entier***
  Cet attribut en lecture seule identifie l'ancienne valeur de l'`ENTIER` de l'attribut modifié de l'objet géré. Le type signé ou non signé correspond à celui de l'attribut modifié.
- ***Nouvelle chaîne***
  Cet attribut en lecture seule identifie la nouvelle valeur de la `CHAÎNE D'OCTETS` dans une notification de modification d'attributs.
- ***Ancienne chaîne***
  Cet attribut en lecture seule identifie l'ancienne valeur de la `CHAÎNE D'OCTETS` dans une notification de modification d'attributs.
- ***Nouvel IDO***
  Cet attribut en lecture seule identifie la nouvelle valeur de l'`IDENTIFICATEUR D'OBJET` dans une notification de modification d'attributs.
- ***Ancien IDO***
  Cet attribut en lecture seule identifie l'ancienne valeur de l'`IDENTIFICATEUR D'OBJET` dans une notification de modification d'attributs.

## Enregistrement de modification de l’état sunPlat

La classe Enregistrement de modification de l’état sunPlat ne détaille pas les informations fournies par la classe Enregistrement de modification des valeurs d’attributs sunPlat. Cette classe permet d’indiquer un changement dans l’attribut de l’objet géré qui reflète un aspect de l’*état* de la ressource.

# PIÈCE 2 Installation et configuration

CHAPITRE 9

# Composants du logiciel de gestion

Ce chapitre décrit les composants qui forment le logiciel de gestion de la plate-forme Sun Fire B1600 et établit la liste des conditions requises pour installer le logiciel SNMP.

Le chapitre contient les sections suivantes :



## Options de gestion du système

Les options de gestion du système suivantes sont décrites pour la plate-forme Sun Fire B1600:

- Surveillance et contrôle du système à l'aide de SNMP
- Fonctionnalité SNMPv3 prenant en charge la gestion sécurisée.
- Surveillance du système à l'aide de Sun Management Center 3.0[1]

1. Pour une description détaillée, notamment de l'installation et de la configuration, reportez-vous au *Sun Management Center 3.0 Supplement for the Sun Fire B1600* (réf. prod. 817-1011-10)

# Instrumentation

En fonction du type de plate-forme, vous pouvez utiliser :

- Un agent de domaine, exécuté sur la lame Sun Fire B100s (surveillance du domaine matérielle)

  Le logiciel est installé localement sur le serveur surveillé et seul ce serveur peut être surveillé. Dans le cas de la plate-forme Sun Fire B1600, chaque lame est surveillée séparément et une seule lame peut être visualisée par chaque instance de l'agent.

- Un agent de plate-forme, mandaté par un contrôleur de système (surveillance de la plate-forme matérielle)

  Le logiciel est installé sur un serveur distant qui accède à la plate-forme par le biais du contrôleur de système. Ceci vous permet de surveiller tout le matériel géré par le contrôleur du système. Dans le cas de la plate-forme Sun Fire B1600, une étagère complète de lames peut être surveillée, y compris les lames de tout type, les sources d'alimentation et les contrôleurs du système.

Dans la FIGURE 9-1, une surveillance de plate-forme matérielle est utilisée pour les étagères A et B de la plate-forme Sun Fire B1600, et une surveillance de domaine matérielle est utilisée pour l'étagère C de la plate-forme Sun Fire B1600.

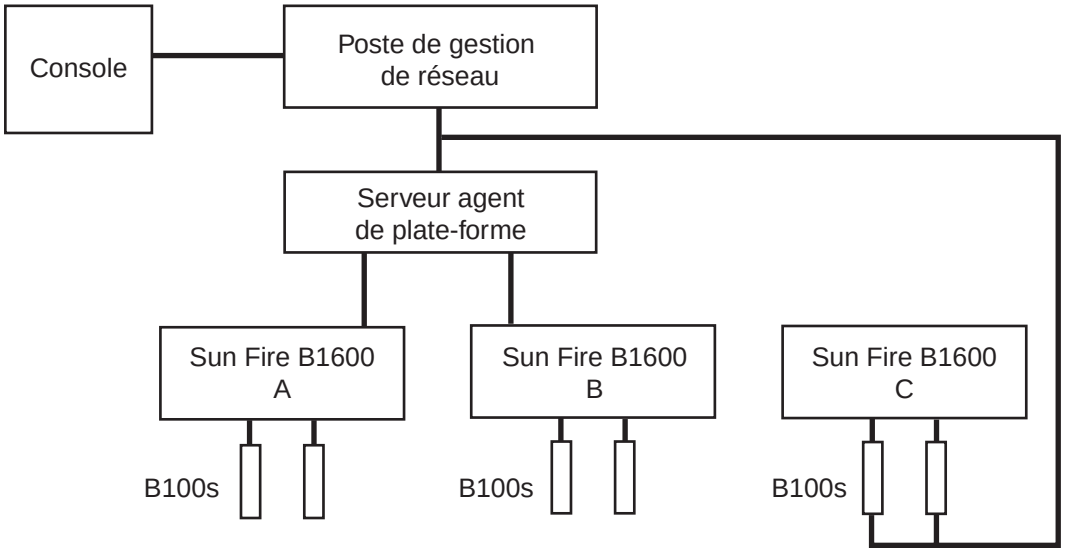


**FIGURE 9-1** Exemple de surveillance de domaine et de plate-forme matérielle

# Configuration système requise

Avant d'installer SNMP Management Agent, assurez-vous que le système est conforme aux conditions préalables requises et aux dépendances abordées dans cette section.

## Système d'exploitation

Le logiciel SNMP Management Agent requiert Solaris 8 Mise à jour 3 ou supérieure.

## Espace disque requis

Au moins 512 Mo doivent être disponibles sur le serveur agent de la plate-forme (1.0 Go recommandé).

## Patch

Les patch suivants doivent être installés, outre le logiciel du système d'exploitation Solaris standard :

### Solaris 8

Aucun patch n'est requis pour la lame Sun Fire B100s.

Java 1.4 doit être installé sur le serveur agent de la plate-forme et la lame (pour la surveillance de domaine et de plate-forme matérielle) avant d'installer le logiciel SNMP Management Agent (voir « Environnement Java », page 66).

### Solaris 9

Aucun patch n'est requis.

# Environnement Java

Pour surveiller entièrement une lame Sun Fire B100s, vous devez préalablement installer les composants Java J2SE 1.4 sur chaque lame Sun Fire B100s surveillée et sur le serveur agent de la plate-forme.

---

**Remarque –** Cette installation met à niveau les logiciels J2SE existants. Si vous ne souhaitez pas mettre le logiciel à niveau (par exemple, parce que vous disposez d'applications ayant été approuvées par la version J2SE 1.3.1 par défaut), vous pouvez installer l'environnement J2RE pour qu'il co-existe avec le système par défaut J2SE. Ceci requiert une configuration supplémentaire du logiciel Sun SNMP Management Agent, décrite dans l'annexe A.

---

Si vous surveillez uniquement la plate-forme Sun Fire B1600 sans l'instrumentation de la cible, vous ne devez installer préalablement les composants Java J2SE 1.4 que sur le serveur agent de la plate-forme. Dans ce cas, l'instrumentation pour le disque dur, les informations sur l'UC et l'adresse MAC Ethernet n'est pas disponible.

Pour s'assurer que les fichiers Java 1.4 sont installés à l'emplacement correct (`/usr/j2se`), utilisez le module `j2sdk-1_4_0_03-solaris-sparc.tar.Z` pour les installer.

Le fichier est accessible depuis

`http://java.sun.com/j2se/1.4/download.html`

Sélectionnez le fichier de téléchargement de SK pour `Solaris SPARC 32-bit tar.Z`

Suivez les instructions de téléchargement qui sont disponibles à l'emplacement indiqué ci-dessus.

---

**Remarque –** Ce nom de fichier est correct au moment de l'écriture. Assurez-vous que vous possédez la dernière version de ce fichier. Le nom du fichier possède le format `j2sdk-1_4_0_<ver>-solaris-sparc.tar.Z,`, où `<ver>` correspond à la révision du logiciel.

---

Cette installation remplaçant le système J2SE, pour vous assurer que les applications Java existantes continuent de fonctionner correctement, vous devez également installer les modules J2SE 1.4 64 bits, contenus dans le fichier `j2sdk-1_4_0_<ver>-solaris-sparcv9.tar.Z`.

---

**Précaution –** J2SE 1.4 est destiné à remplacer J2SE 1.3.1 sous Solaris 8. Vous devez donc désinstaller ce dernier avant d'installer J2SE 1.4. Si vous installez une mise à jour trimestrielle ultérieure pour Solaris 8, certains des modules J2SE 1.4 seront écrasés par les modules J2SE 1.3.1. Pour vous assurer que J2SE 1.4 est installé à l'emplacement correct, utilisez `pkgadd` pour l'installer.

---

### Confirmation de l’installation

Pour vous assurer que l’installation est correcte, utilisez la commande suivante :

```
# /usr/j2se/bin/java -version
java version « 1.4.1_03 »
Java(TM) 2 Runtime Environment, Édition standard (version
1.4.1_03-b04)
Java HotSpot(TM) Client VM (version 1.4.1_03-b04, mode mixte)
```

Cette commande indique la version installée sur le système.

## API SNMP Java

Les modules d’installation incluent une version plus récente de l’API SNMP Java, `SUNWjsnmp`. Supprimez la version existante de ce module à l’aide de la commande `pkgrm` avant d’installer le logiciel Management Agent.

# Modules d’installation

Les modules comprenant le logiciel de gestion peuvent être divisés en groupes de la manière suivante :

- Modules requis pour la surveillance du domaine matérielle
- Modules requis pour la surveillance de la plate-forme matérielle sur le serveur agent de la plate-forme
- Modules requis pour la surveillance de la plate-forme matérielle sur la machine cible

Ces modules sont indiqués dans la TABLEAU 9-1.

**TABLEAU 9-1** SNMP Management Agent Descriptions des modules logiciels

| Module | Nom du module | Fonction |
|---|---|---|
| SUNWbgpc | Cadre du module SAP Personality | Cadre prenant en charge les modules Personality |
| SUNWbgptk | Kit d'outils du module SPA Personality | Bibliothèque réutilisable de modèles de composants et bibliothèques d'accès aux données |
| SUNWbgpr | Module SPA Personality (racine) | Script de démarrage RDP |
| SUNWbgcm | Gestionnaire d'objets de la plate-forme HW SPA | Gestionnaire d'objets de la plate-forme |
| SUNWbgcmr | Gestionnaire d'objets de la plate-forme HW SPA (racine) | Script de démarrage du gestionnaire d'objets de la plate-forme |
| SUNWbgpm | Agent maître / médiateur du protocole SNMP SPA | Agent maître et prise en charge du protocole SNMP |
| SUNWbgpmr | Agent maître / médiateur du protocole SNMP SPA (racine) | Script de démarrage des composants SNMP |
| SUNWbgidr | Détection de domaine SPA (racine) | Script de démarrage de détection de l'agent de domaine |
| SUNWbgod | Détection de plate-forme SPA | Démon de détection de l'agent de plate-forme |
| SUNWbgodr | Détection de plate-forme SPA (racine) | Script de démarrage de détection de l'agent de plate-forme |
| SUNWbgpji | Module SPA Sun Fire B100s Domain Personality | Instrumentation du domaine B100s |
| SUNWbgpjo | Module SPA Sun Fire B1600 Platform Personality | Instrumentation de la plate-forme B1600 |

**Remarque –** Des dépendances internes existent entre les modules et doivent être installées dans un ordre spécifique (voir « Installation du logiciel SNMP », page 73.

# Mise à niveau du logiciel

Pour mettre le logiciel à niveau, vous devez supprimer le logiciel existant avant de réinstaller la nouvelle version (voir Chapitre 13).

# Livraison des modules

Les modules sont livrés dans des bundles d'archives tar. La TABLEAU 9-2 indique le contenu du bundle d'archives `SUNWspa.1.0.tar.Z` contenant les modules qui assurent la prise en charge de SNMP pour la gestion d'une plate-forme Sun Fire B1600 et la gestion des domaines d'une plate-forme Sun Fire B100s.

Lorsqu'ils sont décompressés, les modules sont situés dans les répertoires indiqués dans la table. Reportez-vous au chapitre 10 pour des instructions détaillées sur l'installation du logiciel.

---

**Remarque –** Assurez-vous que vous possédez la dernière version de ce fichier.

---

**TABLEAU 9-2** SNMP Management Agent Bundle de modules

| Bundle | Description | Contenu |
|---|---|---|
| `SUNspa.1.0.22.tar.Z` | Modules de l'agent de la plate-forme SNMP à installer sur le serveur agent de la plate-forme | `platform/proxy/SUNWbgpc`<br>`platform/proxy/SUNWbgpk`<br>`platform/proxy/SUNWbgcm`<br>`platform/proxy/SUNWbgcmr`<br>`platform/proxy/SUNWbgod`<br>`platform/proxy/SUNWbgodr`<br>`platform/proxy/SUNWbgpjo`<br>`platform/proxy/SUNWbgpm`<br>`platform/proxy/SUNWbgpmr`<br>`platform/proxy/SUNWjdrt`<br>`platform/proxy/SUNWjsnmp` |
| | Modules d'instrumentation SNMP à installer sur la lame Sun Fire B100s | `platform/target/SUNWbgpc`<br>`platform/target/SUNWbgptk`<br>`platform/target/SUNWbgpr`<br>`platform/target/SUNWbgcm` |
| | Modules de l'agent du domaine SNMP à installer sur la lame Sun Fire B100s | `domain/SUNWbgpc`<br>`domain/SUNWbgptk`<br>`domain/SUNWbgcm`<br>`domain/SUNWbgcmr`<br>`domain/SUNWbgidr`<br>`domain/SUNWbgpji`<br>`domain/SUNWbgpm`<br>`domain/SUNWbgpmr`<br>`domain/SUNWjdrt`<br>`domain/SUNWjsnmp` |

Pour décompresser le fichier tar, saisissez :

```
$ zcat SUNWspa.1.0.tar.Z | tar xf -
```

## Installation des modules du domaine ou de la cible sur la plate-forme Sun Fire B100s

Les modules du domaine (dans le cas d'une surveillance de domaine matérielle) ou de la cible (dans le cas d'une surveillance de plate-forme matérielle) doivent être installés sur les lames Sun Fire B100s. Il existe plusieurs manières d'effectuer cette opération, parmi lesquelles :

- Décompression des modules du domaine ou de la cible sur chaque lame Sun Fire B100s
- Décompression des modules du domaine ou de la cible dans un répertoire partagé visible de l'utilisateur racine de chaque lame Sun Fire B100s sur laquelle le logiciel doit être installé
- Configuration des modules du domaine ou de la cible pour l'installation en réseau

# Effet sur les fichiers système

Plusieurs nouveaux fichiers de démarrage sont créés dans `/etc/init.d`, comme indiqué dans la table TABLEAU 9-3, qui est reliée à `/etc/rc<n>.d`.

**TABLEAU 9-3** Scripts de démarrage

| Composant | Démarrage Script | Module Nom | Description du module |
|---|---|---|---|
| Gestionnaire des objets de la plate-forme | `spapom` | `SUNWbgcmr` | Gestionnaire des objets de la plate-forme (racine) |
| Module de détection du domaine matérielle | `spaibdm`* | `SUNWbgidr` | Module de détection du domaine matérielle (racine) |
| Modules externes de données distantes | `spardp`† | `SUNWbgpr` | Module Personality (racine) |
| Agent maître / médiateur du protocole SNMP | `spama` | `SUNWbgpmr` | Médiateur de protocole SNMP Agent maître SNMP (racine) |

* Sur le serveur-lame Sun Fire B100s, uniquement avec des modules de domaine.

† Sur le serveur-lame Sun Fire B100s, uniquement avec des modules de plate-forme / cible.

Le module Détection est automatiquement lancé par `inetd` et une nouvelle entrée est créée dans le fichier `/etc/inetd.conf`.

Le gestionnaire d'objets de la plate-forme surveille l'activité d'un port IP pour les requêtes de ses clients. Ces ports sont enregistrés dans le fichier `/etc/services`.

CHAPITRE **10**

# Installation

Ce chapitre décrit comment installer le logiciel de gestion sur la plate-forme Sun Fire B1600.

Le chapitre contient les sections suivantes :



## Sélection de l'installation

Deux éléments sont à prendre en compte lorsque vous évaluez la configuration du logiciel requise :

1. Configuration de l'instrumentation
2. Configuration de l'interface de gestion

### Configuration de l'instrumentation

En fonction du type de plate-forme, vous pouvez utiliser :

- Un agent de domaine, exécuté sur le système surveillé

  Le logiciel est installé localement sur la lame Sun Fire B100s actuellement surveillée (domaine). Chaque lame est surveillée séparément et une seule lame peut être visualisée à la fois. Il s'agit de la *surveillance du domaine matérielle*.

- Un agent de plate-forme, mandaté par un contrôleur de système

  Le logiciel est installé sur serveur (agent de plate-forme) distant qui accède aux instructions de la plate-forme au moyen du contrôleur du système Sun Fire B1600, et sur les lames (cibles) Sun Fire B100s à surveiller. Ceci vous permet de surveiller l'ensemble du matériel géré par le contrôleur du système, notamment les sources d'alimentation, les contrôleurs du système et toutes les lames. Il s'agit de la *surveillance de la plate-forme matérielle*.

  Si vous utilisez une surveillance de plate-forme matérielle, vous devez installer des modules cibles sur chaque lame Sun Fire B100s pour obtenir des informations sur les disques durs, les UC et les adresses Mac Ethernet.

---

**Remarque –** Le serveur-lame Sun Fire B100s est désignée par le terme *domaine* dans la surveillance du domaine matérielle et par le terme *cible* dans la surveillance de la plate-forme matérielle.

---

Voir également « Instrumentation », page 64.

# Configuration de l'interface de gestion

Le logiciel de gestion est conçu pour être déployé de plusieurs manières. Actuellement, les méthodes suivantes sont prises en charge :

- SNMP utilisant `snmpdx` sans l'agent maître (installation par défaut)
- SNMP utilisant l'agent maître et `snmpdx`

  Dans ce cas, vous devez configurer manuellement l'installation, comme indiqué au Chapitre 11 et Chapitre 12.

# Installation du logiciel SNMP

Cette section résume la procédure à suivre pour installer le logiciel de surveillance. Le processus détaillé pour chaque type d'installation est décrit dans les sections suivantes de ce chapitre.

Avant d'installer le logiciel, assurez-vous que :

- la version requise de Solaris est installée sur le domaine ou la cible (Sun Fire B100s) et le serveur agent de la plate-forme (voir « Système d'exploitation », page 65) ;
- vous avez installé tous les patch nécessaires (voir « Patch », page 65) et les autres modules principaux ne faisant pas partie du logiciel SNMP (voir « Environnement Java », page 66) ;
- vous avez installé Java 1.4, soit en mettant à niveau l'environnement J2SE existant, soit en installant un environnement J2RE distinct, comme indiqué dans la rubrique « Environnement Java », page 66.

Lorsque vous êtes certain que le système satisfait à tous ces critères, vous pouvez procéder à l'installation du logiciel SNMP.

Vous devez maintenant choisir entre l'utilisation d'une surveillance de domaine matérielle ou d'une surveillance de plate-forme matérielle.

- Si vous optez pour une surveillance de domaine matérielle, observez la procédure décrite dans la rubrique « Installation du logiciel pour une surveillance de domaine », page 73.
- Si vous optez pour une surveillance de plate-forme matérielle, observez la procédure décrite dans la rubrique « Installation du logiciel pour une surveillance de plate-forme matérielle », page 75.

## Installation du logiciel pour une surveillance de domaine

Installez ces modules sur chaque lame à surveiller (voir « Installation des modules du domaine ou de la cible sur la plate-forme Sun Fire B100s », page 70).

### ▼ Pour installer le logiciel

1. **Assurez-vous que Java 1.4 est installé.**

   Voir « Environnement Java », page 66.

2. **Assurez-vous que les versions existantes de** `SUNWjsnmp` **ont été supprimées.**

   Voir « API SNMP Java », page 67.

3. **Installez les modules de l'agent de domaine sur les lames Sun Fire B100s dans l'ordre indiqué pour éviter des problèmes de dépendance :**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgidr \
SUNWbgpji SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Configurez l'environnement Java.**

   a. **Si vous avez installé J2SE 1.4, comme indiqué dans la rubrique « Environnement Java », page 66, ignorez cette étape.**

   b. **Si vous avez installé J2RE 1.4, comme indiqué dans la rubrique « Installation de J2RE 1.4 », page 121, modifiez les scripts de démarrage de la surveillance de domaine matérielle, comme indiqué dans la section « Surveillance du domaine matérielle », page 123.**

5. **Configurez le logiciel.**

   Voir Chapitre 11.

6. **Redémarrez les serveurs-lames Sun Fire B100s.**

   Vous avez maintenant terminé l'installation du logiciel pour une surveillance de domaine matérielle. Poursuivez à l'aide des « Options d'interface », page 80.

7. **Vérifiez que les processus ont été correctement lancés en saisissant :**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
```

   Si la sortie est identique à celle énoncée ci-dessus, les processus sont correctement exécutés.

# Installation du logiciel pour une surveillance de plate-forme matérielle

- Déterminez si vous souhaitez installer Java 1.4 sur les lames Sun Fire B100s (voir l'énoncé de la section « Environnement Java », page 66). L'installation de Java 1.4 est essentielle si vous souhaitez prendre en charge l'instrumentation de la cible.
  - Pour installer le logiciel et l'instrumentation de la cible, effectuez l'installation indiquée dans la rubrique « Pour installer le logiciel avec l'instrumentation de la cible », page 75.
  - Pour installer le logiciel sans l'instrumentation de la cible, effectuez l'installation indiquée dans la rubrique « Pour installer le logiciel sans instrumentation de la cible », page 77.
- Installez les modules de l'agent plate-forme sur le serveur agent de la plate-forme.
- Installez les modules de l'agent plate-forme sur chaque lame à surveiller, le cas échéant.
- Configurez l'adresse IP SMS du contrôleur du système.

## ▼ Pour installer le logiciel avec l'instrumentation de la cible

1. **Assurez-vous d'avoir installé Java 1.4 sur le serveur agent de la plate-forme.**

   Voir « Environnement Java », page 66 et Step 4 ci-dessous.

2. **Assurez-vous que les versions existantes de** `SUNWjsnmp` **ont été supprimées du serveur agent de la plate-forme.**

   Voir « API SNMP Java », page 67.

3. **Installez les modules de l'agent plate-forme sur le serveur correspondant dans l'ordre indiqué pour éviter tout problème de dépendance :**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Configurez l'environnement Java.**

   a. **Si vous avez installé J2SE 1.4, comme indiqué dans la rubrique « Environnement Java », page 66, ignorez cette étape.**

   b. **Si vous avez installé J2RE 1.4, comme indiqué dans la rubrique « Installation de J2RE 1.4 », page 121, modifiez les scripts de démarrage de la surveillance de la plate-forme matérielle, comme indiqué dans la section « Surveillance de la plate-forme matérielle », page 124.**

5. **Configurez le logiciel.**

   Voir Chapitre 11.

6. **Lancez manuellement l'agent de la plate-forme en saisissant :**

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.snmpdx start
# pkill -1 inetd
```

ou en redémarrant le serveur agent de la plate-forme.

7. **Vérifiez que les processus ont été correctement lancés en saisissant :**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# # netstat -a | grep mismi
     *.mismi            *.*             0     0 24576     0 LISTEN
     *.mismi                            *.*  0
0 24576      0 LISTEN
#
```

Si la sortie est identique à celle énoncée ci-dessus, les processus sont correctement exécutés.

8. **Vérifiez que vous avez installé Java 1.4 sur les lames Sun Fire B100s de la cible actuellement surveillées.**

Voir « Environnement Java », page 66 et Step 11 ci-dessous.

9. **Assurez-vous que les versions existantes de** `SUNWjsnmp` **ont été supprimées des lames de la cible.**

Voir « API SNMP Java », page 67.

10. **Installez les modules de l'agent plate-forme de la cible sur les lames de la cible.**

Ces modules permettent d'accéder aux données d'instrumentation à l'aide des interfaces Solaris de la machine surveillée.

Installez ces modules dans l'ordre indiqué afin d'éviter les problèmes de dépendance.

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgpr
```

11. **Configurez l'environnement Java.**

    a. **Si vous avez installé J2SE 1.4, comme indiqué dans la rubrique « Environnement Java », page 66, ignorez cette étape.**

    b. **Si vous avez installé J2RE 1.4, comme indiqué dans la rubrique « Installation de J2RE 1.4 », page 121, modifiez les scripts de démarrage de la surveillance de la cible matérielle, comme indiqué dans la section « Surveillance de la plate-forme matérielle », page 124.**

12. **Lancez manuellement l'instrumentation de la cible en saisissant :**

```
# /etc/init.d/spama start
```

    ou en redémarrant les systèmes.

13. **Vérifiez que les processus ont été correctement lancés en saisissant :**

```
# netstat -an | grep 1099
      *.1099        *.*          0      0 24576      0 LISTEN
```

    Si la sortie est identique à celle énoncée ci-dessus, le processus est correctement exécuté.

14. **Définissez l'adresse IP de SMS à l'aide de la commande** `setupsc`**.**

    Poursuivez à l'aide de « Configuration du contrôleur du système », page 79.

### ▼ Pour installer le logiciel sans instrumentation de la cible

1. **Assurez-vous d'avoir installé Java 1.4 sur le serveur agent de la plate-forme.**

   Voir « Environnement Java », page 66.

2. **Assurez-vous que les versions existantes de** `SUNWjsnmp` **ont été supprimées du serveur agent de la plate-forme.**

   Voir « API SNMP Java », page 67.

3. **Installez les modules de l'agent plate-forme sur le serveur correspondant dans l'ordre indiqué pour éviter tout problème de dépendance :**

```
# pkgadd -d . SUNWbgptk SUNWbgpc SUNWbgcm SUNWbgcmr SUNWbgod \
SUNWbgodr SUNWbgpjo SUNWjsnmp SUNWjdrt SUNWbgpm SUNWbgpmr
```

4. **Configurez l'environnement Java.**

   a. **Si vous avez installé J2SE 1.4, comme indiqué dans la rubrique « Environnement Java », page 66, ignorez cette étape.**

   b. **Si vous avez installé J2RE 1.4, comme indiqué dans la rubrique « Installation de J2RE 1.4 », page 121, modifiez les scripts de démarrage de la surveillance de la plate-forme matérielle, comme indiqué dans la section « Surveillance de la plate-forme matérielle », page 124.**

5. **Configurez le logiciel.**

   Voir Chapitre 11.

6. **Lancez manuellement l'agent de la plate-forme en saisissant :**

```
# /etc/init.d/spapom start
# /etc/init.d/init.snmpdx stop
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
# pkill -1 inetd
```

   ou en redémarrant le serveur agent de la plate-forme.

7. **Vérifiez que les processus ont été correctement lancés en saisissant :**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
#
# ps -ef | grep spa.wbem

    root   278     1  0   Feb 24 ?       44:19 /usr/j2se/bin/java
    -Dcom.sun.spa.wbem.pomi.port=3333 -Xms64m -Xmx768m -Dcom.sun
#
# # netstat -a | grep mismi
     *.mismi            *.*             0     0 24576     0 LISTEN
     *.mismi                           *.*  0
0 24576      0 LISTEN
#
```

   Si la sortie est identique à celle énoncée ci-dessus, les processus sont correctement exécutés.

8. **Définissez l'adresse IP de SMS à l'aide de la commande** `setupsc`**.**

   Poursuivez ci-dessous.

# Configuration du contrôleur du système

Après avoir installé le logiciel SNMP, vous devez aligner l'adresse IP de SMS dans le contrôleur du système sur celle du serveur agent de la plate-forme. Pour cela, connectez-vous à la console du contrôleur du système, exécutez la commande `setupsc` et ajoutez l'adresse IP du serveur agent de la plate-forme.

Dans l'exemple ci-dessous, l'adresse IP est définie sur `10.5.1.1`.

Appuyez sur [ENTRÉE] après chaque question pour accepter la valeur actuelle jusqu'à ce que la ligne suivante s'affiche :

```
Saisissez l'adresse IP de SMS
```

Saisissez l'adresse IP et appuyez sur [ENTRÉE], puis continuez à appuyer sur [ENTRÉE] pour répondre aux autres questions.

---

**Remarque –** La commande `setupsc` est décrite dans le *Sun Fire B1600 Blade System Chassis Software Setup Guide*.

---

**EXEMPLE DE CODE 10-1** Configuration de l'adresse IP de SMS

```
hornet-sc>setupsc
Mode de configuration interactive initié.
Appuyez sur Ctrl-z pour quitter et enregistrer. Appuyez sur Ctrl-c pour
interrompre l'opération.
Voulez-vous configurer les interfaces activées [o] ?
L'interface réseau SC doit-elle être activée [o] ?
L'interface telnet SC doit-elle être activée pour de nouvelles connexions [o] ?
Voulez-vous configurer l'interface réseau [o] ?
L'interface SC doit-elle utiliser DHCP pour obtenir sa configuration réseau [n]
?
Saisissez l'adresse IP de SC [129.156.174.140] :
Saisissez le masque de sous-réseau IP de SC [255.255.255.0] :
Saisissez la passerelle IP de SC [129.156.174.1] :
Voulez-vous configurer les adresses privées de SC [o] ?
Saisissez l'adresse privée IP SSC0/SC [129.156.174.118] :
Saisissez l'adresse privée IP SSC1/SC [129.156.174.128] :
Voulez-vous activer un VLAN pour le SC [n] ?
Saisissez l'adresse IP de SMS [0.0.0.0] : 10.5.1.1
<tronquée>

hornet-sc>
```

# Options d'interface

L'installation par défaut permet une gestion dans SNMP qui agit comme un sous-agent de la commande `snmpdx`. Aucune saisie utilisateur n'est requise durant l'installation, bien que vous puissiez personnaliser le déploiement après la configuration.

En modifiant les fichiers de configuration, vous pouvez ajouter la fonctionnalité Agent maître à SNMP et la commande `snmpdx`.

---

**Remarque –** Dans tous les cas, la configuration par défaut des listes de contrôle d'accès SNMP ou Access Control Lists (ACL) limite l'accès. Vous devez configurer ces listes afin qu'elles autorisent l'accès (voir Chapitre 11).

---

## SNMP utilisant `snmpdx` (paramètre par défaut)

Cette option enregistre le médiateur SNMP comme un sous-agent de la commande `snmpdx`, en utilisant un numéro de port UDP automatiquement affecté, numéro auquel toutes les requêtes du médiateur sont envoyées. Ces requêtes peuvent être effectuées via `snmpdx`, ou directement, comme l'indique la ligne pointillée dans la FIGURE 10-1, si ce paramètre est activé dans le fichier ACL du médiateur (voir Chapitre 12).

**TABLEAU 10-1** Résumé des ports pour la FIGURE 10-1

| Touche | Fonction | Paramètre dans `spama.conf` | Paramètre par défaut |
|---|---|---|---|
| 1 | Port via lequel la commande `snmpdx` envoie les requêtes au médiateur SNMP | `SPAPM_REQ_PORT` | |
| 2 | Port via lequel le médiateur envoie des traps aux gestionnaires de SNMP | `SPAPM_TRAP_PORT` | 162 |

---

**Remarque –** Bien que la configuration par défaut soit automatique, vous devez toujours configurer les fichiers ACL pour la commande `snmpdx` et/ou le médiateur afin qu'il prenne en charge la configuration du gestionnaire.

---

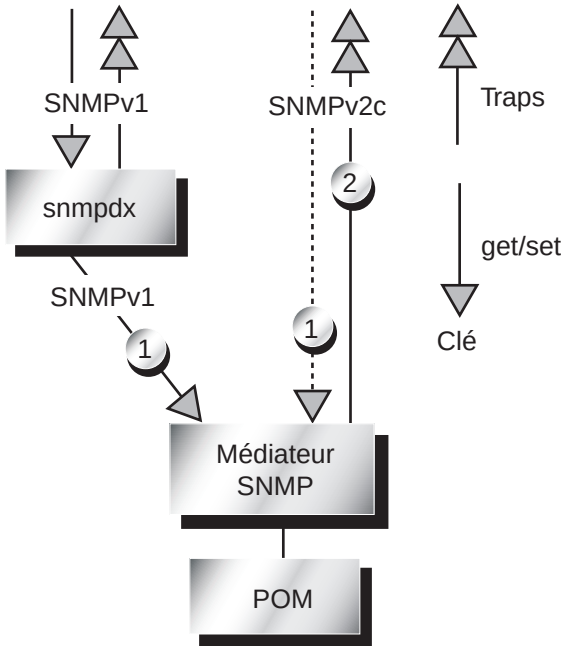


**FIGURE 10-1** Flux de données lorsque SNMP est un sous-agent de `snmpdx`

## SNMP et agent maître associés à la commande `snmpdx`

Cette option ajoute la fonctionnalité de sécurité SNMPv3 de l'agent maître à SNMP et `snmpdx`. Le lancement automatique de la commande snmpdx est désactivé et l'agent maître est enregistré sur le port 161. Un nouveau numéro de port est automatiquement affecté à snmpdx.

Les traps du médiateur sont optionnellement convertis par l'agent maître en SNMPv3 ou envoyés directement.

Les traps de la commande `snmpdx` ne sont envoyés directement qu'aux gestionnaires de SNMP et ne peuvent pas être convertis par l'agent maître.

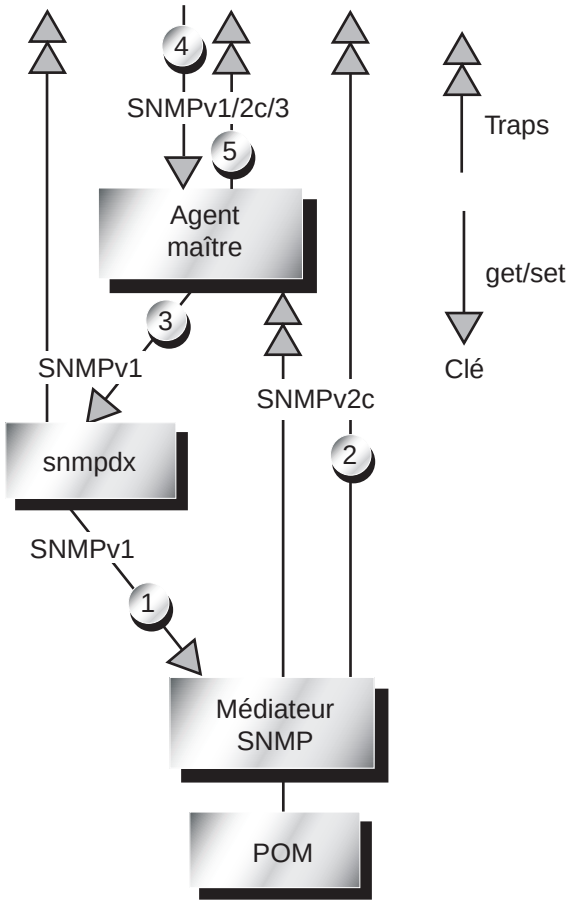


**FIGURE 10-2** Flux de données lorsque l'agent maître est utilisé

**TABLEAU 10-2** Résumé des ports pour la FIGURE 10-2

| Touche | Fonction | Paramètre dans `spama.conf` | Paramètre par défaut |
|---|---|---|---|
| 1 | Port via lequel la commande `snmpdx` envoie les requêtes au sous-agent du médiateur SNMP | `SPAPM_REQ_PORT` | |
| 2 | Port via lequel le médiateur envoie des traps directement aux gestionnaires de SNMP | `SPAPM_TRAP_PORT` | 162 |
| 3 | Port via lequel l'agent maître envoie des requêtes à la commande `snmpdx` | `SNMPDX_REQ_FORWARD_PORT` | |
| 4 | Port sur lequel les requêtes sont surveillées par l'agent maître | `MASTER_AGENT_REQ_PORT` | 161 |
| 5 | Port via lequel l'agent maître envoie des traps aux gestionnaires de SNMP | `SPAPM_TRAP_PORT` | 162 |

# Agent maître tiers et SNMP

Cette option enregistre le médiateur SNMP comme un sous-agent utilisant un numéro de port affecté manuellement ou par l'agent maître tiers. Pour activer l'accès direct, vous devez configurer manuellement le fichier ACL du médiateur, comme indiqué au Chapitre 12.

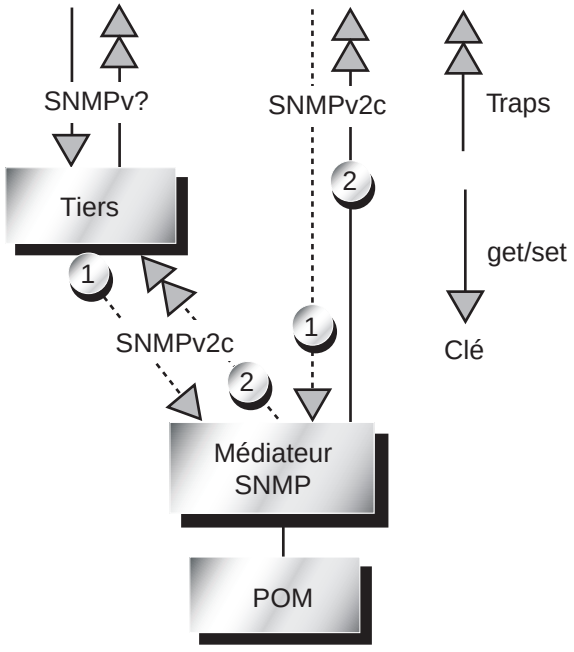


**FIGURE 10-3** Flux de données lorsqu'un agent maître tiers est utilisé

**TABLEAU 10-3** Résumé des ports pour la FIGURE 10-3

| Touche | Fonction | Paramètre dans `spama.conf` | Paramètre par défaut |
|---|---|---|---|
| 1 | Port utilisé par le médiateur SNMP permettant un accès direct ou sur lequel l'agent maître tiers envoie des requêtes au médiateur SNMP | `SPAPM_REQ_PORT` | |
| 2 | Port utilisé par le médiateur SNMP pour envoyer des traps SNMPv2c à l'agent maître tiers ou directement aux gestionnaires de SNMP | `SPAPM_TRAP_PORT` | |

CHAPITRE **11**

# Fichiers de configuration

Ce chapitre contient une vue d'ensemble des fichiers que vous pouvez modifier pour configurer le logiciel. Il dresse la liste des paramètres configurables et présente le concept du contrôle d'accès.

Lisez ce chapitre avant de consulter le Chapitre 12 qui explique comment configurer les principales options SNMP à l'aide des fichiers décrits ici.

Ce chapitre contient les sections suivantes :



**Remarque –** Pour de plus amples informations sur les modules SNMP et leur installation, reportez-vous au Chapitre 9 et Chapitre 10.

# Fichiers de configuration

Les fichiers suivants, situés dans `/etc/opt/SUNWspa/`, déterminent la configuration de SNMP :

- **Fichier de configuration générale**
  - `spama.conf`—Définit le mode de configuration de l'agent maître et du médiateur
- **Fichiers de configuration du médiateur**
  - `spapm.acl`—Définit le contrôle d'accès pour le médiateur
  - `spapm_snmpdx.acl`—Définit le contrôle d'accès pour le médiateur exécuté comme sous-agent de `snmpdx`
- **Fichiers de configuration de l'agent maître**

  Si vous n'utilisez pas l'agent maître, vous n'avez pas besoin de configurer ces fichiers.
  - `spapm.acl`—Définit le contrôle d'accès pour l'agent maître
  - `spapm.uacl`—Définit l'utilisateur de SNMPv3 et le contrôle d'accès contextuel pour l'agent maître
  - `spama.security`—Définit les utilisateurs de SNMPv3 référencés dans `spama.uacl`

Ces fichiers sont décrits en détail dans les sections suivantes.

# Fichier de configuration générale

## `spama.conf`

Le fichier `spama.conf` contient un certain nombre de paramètres configurables, décrits dans la section suivante et la TABLEAU 11-1. Un exemple de fichier `spama.conf` est illustré dans l'EXEMPLE DE CODE 11-1.

## Options générales

### START_MEDIATOR

Définissez ce paramètre sur `Oui` pour exécuter le médiateur. Sinon, sélectionnez `Non` (voir également FIGURE 10-1).

La valeur par défaut est :

```
START_MEDIATOR=oui
```

### START_MASTER_AGENT

Si vous souhaitez utiliser la sécurité SNMPv3 et que vous ne recourez pas à un agent maître tiers pour l'assurer, définissez ce paramètre sur `Oui` pour lancer l'agent maître (voir la FIGURE 10-2 et la table jointe).

Si vous ne nécessitez pas la sécurité SNMPv3 et vous que recourez à un autre agent maître (notamment snmpdx), ou que vous ne souhaitez utiliser aucun agent maître, définissez ce paramètre sur `Non` (voir également la FIGURE 10-1 et FIGURE 10-3, ainsi que les tables jointes).

La valeur par défaut est :

```
START_MASTER_AGENT=non
```

### AGENT_INTERFACE_NAME

Si l'agent maître est activé (voir ci-dessus), la valeur définie ici indique une interface réseau à laquelle l'agent maître doit être lié, le protocole du médiateur étant lié à l'`hôte local`.

Si l'agent maître n'est pas activé, la valeur ici choisie définit le nom d'hôte de l'interface réseau à laquelle le médiateur est lié. Si aucune valeur n'est spécifiée, l'accès s'effectue par défaut via l'interface par défaut.

Si vous utilisez le médiateur comme sous-agent de `snmpdx`, par exemple, sélectionnez le paramètre `hôte local`.

La valeur par défaut est :

```
AGENT_INTERFACE_NAME=hôte local
```

## Options de l’agent maître

Pour que l’agent maître puisse fonctionner avec la commande `snmpdx`, le script de démarrage de l’agent maître arrête `snmpdx`, prend le relais de son port SNMP et redémarre `snmpdx` comme sous-agent sur un autre port.

Si le paramètre `SNMPDX_REQ_FORWARD_PORT` affiche une valeur `nulle`, le script de démarrage de l’agent maître recherche un port disponible dans la plage anonyme éphémère 32768-65535 et redémarre `snmpdx` comme sous-agent sur ce port. Le script de démarrage parcourt également `/etc/services` et n’utilise aucun des ports qui y sont énumérés.

Cependant, si vous indiquez la valeur `SNMPDX_REQ_FORWARD_PORT`, l’agent maître utilise ce port pour transmettre les requêtes à la commande `snmpdx`, dans quel cas il ne vérifie pas si le port est déjà utilisé.

### MASTER_AGENT_REQ_PORT

Il s’agit du port via lequel l’agent maître reçoit les requêtes des gestionnaires. Pour la plupart des configurations, il n’est pas nécessaire de modifier la valeur (voir également la FIGURE 10-2 et la table jointe).

Si aucune valeur n’est indiquée, la valeur par défaut est `161`.

### ENABLE_SNMPV2C_SETS

Ce paramètre détermine si l’agent maître permet d’effectuer des opérations à l’aide de SNMPv1 ou de SNMPv2c. Si vous sélectionnez l’option `Oui`, la sécurité est substantiellement réduite étant donné que les protocoles SNMPv1 et SNMPv2C sont intrinsèquement peu sûrs.

La valeur par défaut est :

```
ENABLE_SNMPV2C_SETS=non
```

### SNMPDX_REQ_FORWARD_PORT

Ce paramètre contrôle le port via lequel l’agent maître envoie des requêtes à `snmpdx`. Si aucune valeur n’est spécifiée, une configuration automatique est effectuée par l’agent maître (voir introduction de cette section, la FIGURE 10-2 et la table jointe).

Dans le cas contraire, vous devez également configurer manuellement la commande `snmpdx` pour écouter sur ce port.

La valeur par défaut est :

```
SNMPDX_REQ_FORWARD_PORT=
```

### SNMPV3_USER

Ce paramètre détermine quel utilisateur SNMPv3 émet des traps SNMPv3.

La valeur par défaut est :

```
SNMPV3_USER=Utilisateur par défaut
```

---

**Remarque –** Pour envoyer des traps SNMPv3, vous devez définir `SPAPM_TRAPS_ARE_V3=oui`.

---

## Options du médiateur du protocole

### SUB_AGENT

Ce paramètre détermine si le médiateur ou l'agent maître est démarré par un agent maître comme snmpdx, et non automatiquement au démarrage.

La valeur par défaut est :

```
SUB_AGENT=oui
```

Si vous sélectionnez l'option Oui, vous devez démarrer le médiateur en communiquant un paramètre de <`port`> comme suit :

```
# /etc/init.d/spama start <port>
```

où <`port`> correspond au port UDP sur lequel le médiateur écoute les requêtes SNMP. Par exemple, lorsqu'elle est utilisée avec la commande `snmpdx`, l'invocation du médiateur est contrôlée par la ligne suivante dans le fichier `/etc/snmp/conf/spapm.rsrc` :

```
commande = "etc/init.d/spama start $PORT"
```

---

**Remarque –** Si `SUB-AGENT=OUI`, la valeur de `START_MASTER_AGENT` est ignorée. Si vous activez la fonctionnalité Agent maître, vous devez définir `SUB_AGENT` sur `Non`.

---

Si vous spécifiez `Non`, le médiateur est lancé au démarrage (par le script de démarrage `/etc/rc3.d/S80spama`), et le port UDP sur lequel il écoute les requêtes SNMP est défini par le paramètre `SPAPM_REQ_PORT` décrit ci-dessous.

### SPAPM_REQ_PORT

Ce paramètre détermine le port sur lequel le médiateur reçoit les requêtes lorsque SUB_AGENT est défini sur Non. Voir la FIGURE 10-1, FIGURE 10-2 et FIGURE 10-3, ainsi que les tables jointes.

La valeur par défaut est :

SPAPM_REQ_PORT=

Avec le paramètre par défaut, si START_MASTER_AGENT=oui, un numéro de port est automatiquement affecté. Si START_MASTER_AGENT=non, le numéro du port par défaut 33000 est utilisé.

Si START_MASTER_AGENT=non, vous devez définir SPAPM_REQ_PORT sur la valeur désirée, afin que le médiateur soit accessible par un agent maître local à l'aide d'un port fixe, ou directement par le biais de gestionnaires SNMP distants.

### SPAPM_TRAPS_ARE_V3

Ce paramètre détermine si les traps du médiateur sont des traps SNMPv3 ou SNMPv2c.

La valeur par défaut, qui définit les traps dans SNMPv2c, est :

SPAPM_TRAPS_ARE_V3=non

---

**Remarque –** Si vous activez les traps SNMPv3 (SPAPM_TRAPS_ARE_V3=oui), vous devez également définir START_MASTER_AGENT=oui et START_MEDIATOR=oui.

---

### SPAPM_TRAP_PORT

Ce paramètre détermine le numéro du port auquel les traps du médiateur sont envoyés. Voir la FIGURE 10-1, FIGURE 10-2 et FIGURE 10-3, ainsi que les tables jointes.

La valeur par défaut est :

SPAPM_TRAP_PORT=162

### SPAPM_TRAP_INTERFACE

Ce paramètre détermine l'interface à partir de laquelle le médiateur envoie les traps SNMP. S'il n'est pas défini, les traps sont émis depuis l'interface par défaut de l'hôte.

---

**Remarque –** Les traps SNMPv2c sont envoyés directement et ne passent pas par la commande snmpdx.

---

La valeur par défaut est indéfinie :

```
SPAPM_TRAP_INTERFACE=
```

## SPAPM_OPTIONS

Ce paramètre vous permet de modifier le comportement du médiateur en spécifiant une ou plusieurs options suivantes :

- `-a` Envoyer des notifications de modification des attributs
- `-s` Envoyer des notifications de modification de l'état
- `-c` Envoyer des notifications de création d'objets
- `-C` Activer des notifications de création d'objets lors de l'initialisation
- `-c` Envoyer des notifications de suppression d'objets
- `-l` Activer des journaux de listes de problèmes courants par défaut

Le format et la valeur par défaut de ce paramètre sont :

```
SPAPM_OPTIONS="-ascCdl"
```

**TABLEAU 11-1** Valeurs par défaut du fichier `spama.conf`

| Paramètre | Valeur suivant l'installation par défaut |
|---|---|
| START_MEDIATOR | START_MEDIATOR=oui |
| START_MASTER_AGENT | START_MASTER_AGENT=non |
| AGENT_INTERFACE_NAME | AGENT_INTERFACE_NAME=hôte local |
| MASTER_AGENT_REQ_PORT | MASTER_AGENT_REQ_PORT=161 (le paramètre par défaut est adopté si cette valeur n'est pas spécifiée) |
| ENABLE_SNMPV2C_SETS | ENABLE_SNMPV2C_SETS=non |
| SNMPDX_REQ_FORWARD_PORT | SNMPDX_REQ_FORWARD_PORT= |
| SNMPV3_USER | SNMPV3_USER=Utilisateur par défaut |
| SUB_AGENT | SUB_AGENT=oui |
| SPAPM_REQ_PORT | SPAPM_REQ_PORT= |
| SPAPM_TRAPS_ARE_V3 | SPAPM_TRAPS_ARE_V3=non |
| SPAPM_TRAP_PORT | SPAPM_TRAP_PORT=162 |
| SPAPM_TRAP_INTERFACE | SPAPM_TRAP_INTERFACE= |
| SPAPM_OPTIONS | SPAPM_OPTIONS="-ascCdl" |

**EXEMPLE DE CODE 11-1** Exemple de fichier spama.conf

```
#!/sbin/sh
#
#ident  "@(#)spama.conf1.17 01/29/03 SMI"
#
# Copyright 2003 Sun Microsystems, Inc. Tous droits réservés.
# Son utilisation est soumise à des conditions de licence.

#
# Ce fichier permet de contrôler la configuration de l’agent maître et du
# médiateur de protocole
#

#
# Configuration de l’agent maître / médiateur
#

################################################################################
# Options générales
################################################################################

#
# Sélectionnez Oui si le composant médiateur doit être démarré
#
START_MEDIATOR=oui

#
# Sélectionnez Oui pour activer l’agent maître
#
START_MASTER_AGENT=non

#
# Nom d’hôte de l’interface réseau auquel l’agent doit être lié. Si ce nom
# n’est pas spécifié, l’agent est accessible via
# l’interface par défaut.
#
# Si le médiateur est utilisé comme sous-agent, il doit être
# défini sur l’hôte local.
#
# Si l’agent maître est activé, ce paramètre s’applique à son interface,
# le médiateur du protocole étant lié à l’hôte local.
AGENT_INTERFACE_NAME=hôte local

################################################################################
# Options de l’agent maître
################################################################################

#
```

**EXEMPLE DE CODE 11-1** Exemple de fichier spama.conf *(suite)*

```
# Port SNMP permettant à l'agent maître de recevoir des requêtes get/set SNMP.
#
# Ce numéro de port sera utilisé pour écouter les
# du médiateur de protocole non SNMP.
#
# Si cette valeur n'est pas spécifiée, le port par défaut est 161 si
START_MASTER_AGENT=oui
#
MASTER_AGENT_REQ_PORT=

#
# sélectionnez Oui pour autoriser les opérations SET de SNMPv1/SNMPv2c via
l'agent maître.
#
ENABLE_SNMPV2C_SETS=non

#
# Port du sous-agent SNMP auquel les requêtes (snmpdx) du médiateur de protocole
non SNMP
# sont envoyées.
#
# Si le paramètre de ce port n'est pas spécifié (par défaut), une configuration
automatique est
# effectuée. Le numéro du port pour la commande snmpdx est déterminé de manière
dynamique
# si snmpdx utilise déjà le port UDP où l'agent maître écoute
# les requêtes get/set SNMP (port UDP 161 par défaut) au démarrage de l'agent
maître.
#
# Si le paramètre de ce port n'est pas spécifié, mais que snmpdx n'utilise pas
le même port
# étant donné que l'agent maître écoute les requêtes get/set SNMP, l'agent
maître
# utilise le numéro de port utilisé par snmpdx pour envoyer
# les requêtes set/get SNMP à snmpdx.
#
# Si ce port est défini, l'utilisateur doit effectuer
# la configuration du port «à l'écoute » pour les sous-agents. L'agent
# maître utilise alors ce numéro de port pour transmettre des requêtes
# du médiateur de protocole non SNMP.
#
SNMPDX_REQ_FORWARD_PORT=

#
# utilisateur de SNMPv3
#
```

**EXEMPLE DE CODE 11-1** Exemple de fichier spama.conf *(suite)*

```
# Le médiateur fait appel à cet utilisateur pour envoyer des traps V3 (s'il est
activé dans
# SPAPM_TRAPS_ARE_V3).
#
# Si cette valeur n'est pas spécifiée, le paramètre par défaut est Utilisateur
par défaut.
SNMPV3_USER=

##############################################################################
# Options du médiateur de protocole
##############################################################################

#
#
# Configuration du sous-agent
#
# Si l'agent maître n'est pas utilisé (c'est-à-dire START_MASTER_AGENT=non),
alors
# le paramètre SUB_AGENT=oui indique que le médiateur doit être démarré avec un
# argument de numéro de port par snmpdx ou un agent maître tiers. Sinon,
choisissez
# l'option Non si le médiateur doit être lancé avec un numéro de port
manuellement
# configuré.
#
# Si START_MASTER_AGENT=oui, alors ce paramètre est ignoré.
#
SUB_AGENT=oui

#
# Port des requêtes du médiateur. Si START_MASTER_AGENT=non et si SUB_AGENT=
non, le
# paramètre par défaut est 33000. Sinon, ce paramètre est affecté de manière
dynamique.
#
SPAPM_REQ_PORT=

#
# sélectionnez l'option Oui pour activer les traps du médiateur v3 (requiert
que START_MASTER_AGENT=oui et
# START_MEDIATOR=oui)
#
SPAPM_TRAPS_ARE_V3=non

#
# Port des traps par défaut
#
```

**EXEMPLE DE CODE 11-1** Exemple de fichier `spama.conf` *(suite)*

```
SPAPM_TRAP_PORT=162

# Ce paramètre permet également de définir l'interface à laquelle les traps SNMP
sont
# envoyés par le médiateur du protocole, indépendamment de la définition de
# AGENT_INTERFACE_NAME. Si le paramètre n'est pas défini, les traps sont émis
depuis l'
# interface hôte par défaut.
#
SPAPM_TRAP_INTERFACE=
#
# Balises d'options de l'agent
#
# -a    Envoyer des notifications de modification des attributs
# -s    Envoyer des notifications de modification de l'état
# -c    Envoyer des notifications de création d'objets
# -C    Activer les notifications de création d'objets lors de l'initialisation
# -d    Envoyer des notifications de suppression d'objets
# -l    Activer les journaux de listes de problèmes courants par défaut
#
SPAPM_OPTIONS="-ascCdl"
```

# Contrôle de l'accès

Le contrôle de l'accès est basé sur l'adresse IP et la communauté de la machine hôte des gestionnaires et les communautés spécifiées pour chaque sous-agent. Les droits d'accès accordés aux communautés et aux machines hôtes sont définis dans les fichiers ACL.

Les fichiers ACL définissent également les hôtes auxquels l'agent envoie des traps. Lors de l'envoi d'un trap, l'agent le transmet à tous les hôtes répertoriés dans la liste <*trapInterestHostList*> du fichier ACL.

Il existe trois fichiers ACL :

- `spapm.acl` contrôle l'accès au médiateur, soit directement depuis les applications de gestion, soit normalement depuis la commande `snmpdx`. Il définit également les destinataires requis des traps SNMP.
- `spapm_snmpdx.acl` contrôle l'accès au médiateur lorsqu'il est configuré comme sous-agent de `snmpdx`.
- `spama.acl` contrôle l'accès via l'agent maître SNMPv3

Les paramètres de transmission du contrôle d'accès, des communautés et des traps pour SNMPv3 sont définis dans les fichiers `spama.uacl` et `spama.security`.

# Format d'un fichier ACL

Un fichier ACL contient une communauté de définition de groupes `acl` et des droits d'accès aux gestionnaires et un groupe de `traps` définissant la communauté et les hôtes permettant d'envoyer des traps. Un fichier ACL peut également contenir des lignes de commentaires, dont le premier caractère est un dièse (#).

---

**Remarque –** Il existe certaines différences dans la syntaxe du fichier `spapm_snmpdx.acl`. Reportez-vous aux commentaires pour plus de détails.

---

## Le groupe `acl`

Un groupe `acl` contient une ou plusieurs listes de configurations de communautés utilisant la syntaxe suivante :

```
acl = {
    <list1>
    <list2>
    ...
    <listN>
}
```

Le groupe `acl` de ce fichier indique les droits d'accès pour des communautés et des gestionnaires spécifiques. Il comprend une liste de configurations de communautés au format suivant :

```
{
   communautés = <communityList>
   accès = <accessRights>
   gestionnaires = <hostList>
}
```

La liste <*communityList*> contient les noms de communauté SNMP auxquels le contrôle de l'accès s'applique. Les noms de communautés figurant dans cette liste sont séparés par des virgules.

La table <*accessRights*> spécifie les droits accordés à tous les gestionnaires exécutés sur les machines indiquées dans l'élément Managers. Il existe deux valeurs possibles :

- `lecture-écriture`
- `lecture seule`

L'élément <*hostList*> spécifie les machines hôtes des gestionnaires auxquelles des droits d'accès sont accordés. La liste <*hostList*> est une liste d'hôtes séparés par une virgule. Chacun de ces hôtes peut être :

- un nom d'hôte (par exemple, `hubble`) ;
- une adresse IP (par exemple, `123.456.789.12`) ;
- un masque de sous-réseau (par exemple, `123!255!255!255`)

---

**Remarque –** Pour distinguer des adresses IP de masques de sous-réseau dans un fichier ACL, chaque entier d'un masque de sous-réseau est séparé par un point d'exclamation (!) au lieu d'un point.

---

**EXEMPLE DE CODE 11-2** Exemple d'un groupe `acl`

```
acl = {
    {
        communautés = publique, privée
        accès = lecture seule
        gestionnaires = rag, tag, bobtail
    }
    {
        communautés = tigger
        accès = lecture-écriture
        gestionnaires = brittas
    }
}
```

## Le groupe `trap`

Le groupe `trap` spécifie les hôtes auxquels l'agent peut envoyer des traps. La configuration n'est requise que si le médiateur doit envoyer des traps SNMPv2, non s'il doit envoyer des traps SNMPv3 au moyen de l'agent maître SNMP.

Ce groupe contient une ou plusieurs définitions de communautés de traps utilisant la syntaxe suivante :

```
trap = {
    <community1>
    <community2>
   ...
    <communityN>
}
```

Chaque ligne définit l'association entre un ensemble d'hôtes et la chaîne de communauté SNMP des traps qui sont destinés à leur être envoyés.
Chaque définition `trap-communauté` possède le format suivant :

```
{
   trap-communauté = <trapCommunityString>
   hôtes = <trapInterestHostList>
}
```

L'élément <*trapCommunityString*> spécifie la chaîne de communauté SNMP.
Il est inclus dans les traps envoyés aux hôtes spécifiés dans l'élément `hosts`.

L'élément <*trapInterestHostList*> spécifie une liste d'hôtes séparés par des virgules. Chaque hôte doit être identifié par son nom ou adresse IP complète, comme indiqué dans l'exemple suivant :

**EXEMPLE DE CODE 11-3** Exemple d'un groupe `trap`

```
trap = {
     {
          trap-communauté = tigger
          hôtes = gandalf, frodo
     }
}
```

# Fichiers de configuration du médiateur

Cette section décrit le format du :

« Fichier `spapm.acl` », page 99

« Fichier `spapm_snmpdx.acl` », page 101

## Fichier `spapm.acl`

Le fichier `spapm.acl` définit le contrôle de l'accès au médiateur du protocole. Le format général du fichier est décrit à la page « Format d'un fichier ACL », page 96. Cette section contient des informations liées spécifiquement au fichier `spapm.acl`, et un exemple du fichier, comme indiqué dans l'EXEMPLE DE CODE 11-4.

Le fichier est situé par défaut dans `/etc/opt/SUNWspa`.

Si un fichier ACL existe, les droits d'accès qu'il définit s'appliquent à tous les gestionnaires ou serveurs agents de plate-forme qui accèdent à l'agent via son adaptateur SNMP. Si le fichier ACL n'existe pas lorsque les agents sont lancés, tous les gestionnaires sont autorisés à accéder à l'agent via l'adaptateur SNMP et aucun trap n'est généré.

Pour activer le contrôle de l'accès et les traps pour l'adaptateur SNMP, assurez-vous qu'un fichier ACL existe au démarrage des agents. Le fichier ACL contenant des informations sur la sécurité, affectez-lui des droits d'accès restreints, lisibles uniquement par la `racine`.

Les groupes `acl` et `trap` observent les formats décrits dans les rubriques « Le groupe `acl` », page 96 et « Le groupe `trap` », page 98, respectivement.

Si le médiateur est enregistré comme un sous-agent d'un agent maître (tel que `snmpdx`), vous devez spécifier que l'`hôte local` est un gestionnaire dans le fichier `spapm.acl` puisqu'il constitue le point d'origine d'envoi des paquets SNMP par l'agent maître. Lorsque vous utilisez la commande `snmpdx`, elle envoie des chaînes de communauté sans modification et doit, par conséquent, spécifier également dans le fichier `spapm_snmpdx.acl` les communautés qui sont énumérées dans ce fichier (voir EXEMPLE DE CODE 11-4).

**EXEMPLE DE CODE 11-4** Exemple de fichier `spapm.acl`

```
#
#  @(#)spapm.acl     1.6 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. Tous droits réservés.
#
# Modèle de fichier ACL de Sun SNMP Management Agent pour Sun Fire B1600

acl = {
    {
        communautés = publique, privée
        accès = lecture seule
        gestionnaires = rag, tag, bobtail
    }
    {
        communautés = tigger
        accès = lecture-écriture
         gestionnaires = brittas
     }
}

trap = {
    {
        trap-communauté = tigger
        hôtes = brittas
    }
}
```

Le groupe de traps définit l'emplacement auquel les notifications SNMPv2c sont envoyées.

# Fichier spapm_snmpdx.acl

Dans la configuration par défaut, le médiateur est exécuté comme sous-agent de snmpdx. Vous pouvez modifier ce fichier pour autoriser un accès basé sur le nom d'hôte source. Les communautés et les droits d'accès doivent correspondre à ceux du fichier spama.acl.

Le groupe acl de ce fichier indique les droits d'accès pour des communautés et des gestionnaires spécifiques. Il comprend une liste de configurations de communautés au format suivant :

```
#   {
#       communautés = <communityList>
#       accès = <accessRights>
#       gestionnaires = <hostList>
#   }
```

- L'élément *communityList* est une liste de noms de communautés séparés par des virgules auxquels ce contrôle d'accès s'applique.
- L'élément *accessRights* spécifie les autorisations accordées aux gestionnaires désignés dans la liste *hostList*.
- L'élément *hostList* est une liste de noms d'hôtes séparés par une virgule auxquels doivent être accordés les *droits d'accès* spécifiés,

Dans le premier exemple EXEMPLE DE CODE 11-5, les systèmes rag, tag et bobtail sont configurés pour un accès en lecture-écriture sur les communautés publique et privée. Le système brittas est configuré pour un accès en lecture-écriture avec la communauté tigger.

Le deuxième exemple s'applique aux configurations utilisant l'agent maître SNMPv3 (START_MASTER_AGENT=oui dans spama.conf) où les paquets SNMP reçus par snmpdx proviennent de l'hôte local. L'accès en lecture seule est configuré pour les communautés publique et privée. L'accès en lecture-écriture est configuré pour la communauté tigger. Ces communautés sont mappées dans des contextes SNMPv3 par l'agent maître. Par conséquent, tout contexte SNMPv3 pour lequel cet accès est requis doit spécifier une communauté correspondante dans ce fichier.

**EXEMPLE DE CODE 11-5** Exemple de fichier `spapm_snmpdx.acl`

```
#  @(#)spapm_snmpdx.acl1.8 03/01/29 SMI
#
# Copyright 2003 Sun Microsystems, Inc. Tous droits réservés.
# Son utilisation est soumise à des conditions de licence.
#
# Modèle de fichier de contrôle d'accès snmpdx de Sun SNMP Management Agent pour
Sun
# Fire B1600
#
# Exemple 1 :
#
# acl = {
#     {
#         communautés = publique, privée
#         accès = lecture seule
#         gestionnaires = rag, tag, bobtail
#     }
#     {
#         communautés = tigger
#         accès = lecture-écriture
#         gestionnaires = rag, tag, bobtail
#     }
# }
#
# Exemple 2 :
#
 acl = {
     {
         communautés = publique, privée
         accès = lecture seule
         gestionnaires = hôte local
     }
     {
         communautés = tigger
         accès = lecture-écriture
         gestionnaires = hôte local
     }
 }
#
# Les destinations des traps sont définies dans les fichiers spapm.acl et
spama.acl.
# Il n'est pas nécessaire de modifier cette entrée.
trap = {
}
```

# Fichiers de configuration de l'agent maître

Vous ne devez configurer ces fichiers que si la fonctionnalité Agent maître est activée (`START_MASTER_AGENT=oui` dans `spama.conf`).

Cette section décrit le format du :



## Fichier `spama.acl`

Le fichier `spama.acl` définit le contrôle de l'accès pour l'agent maître. Le format général du fichier est décrit à la page « Format d'un fichier ACL », page 96. Cette section contient des informations spécifiquement liées au fichier `spama.acl`.

Le fichier est situé par défaut dans `/etc/opt/SUNWspa`.

Pour activer le contrôle de l'accès et les traps pour l'adaptateur SNMP, assurez-vous qu'un fichier ACL existe au démarrage des agents. Le fichier ACL contenant des informations sur la sécurité, affectez-lui des droits d'accès restreints, lisibles uniquement par la `racine`.

Ce fichier définit les autorisations d'accès à SNMPv1 et SNMPv2c, ainsi que les destinataires des notifications SNMP. Si `SPAPM_TRAPS_ARE_V3=oui` dans le fichier `spama.conf`, les traps sont envoyés comme traps SNMPv3. Sinon, ils sont envoyés comme traps SNMPv2c.

### Groupe `acl`

Si vous utilisez le protocole SNMPv3, vous souhaitez peut-être interdire l'accès en écriture à SNMPv1 et SNMPv2c. Par conséquent, ce fichier `acl` n'autorise généralement qu'un accès en lecture seule.

**EXEMPLE DE CODE 11-6** Exemple d'un groupe `acl`

```
acl = {
     {
          communautés = publique, privée
          accès = lecture seule
          gestionnaire = hôte local
     }
}
```

### Groupe `trap`

Le groupe `trap` de ce fichier observe le format décrit dans la rubrique « Le groupe `trap` », page 98.

## Fichier `spama.uacl`

Ce fichier est utilisé en association avec le fichier `spama.security` pour activer la sécurité SNMPv3 lorsque l'agent maître est actif. Le format général du fichier est décrit dans la rubrique « Format d'un fichier ACL », page 96. Les paramètres de configuration supplémentaires sont expliqués dans cette section. Un exemple du fichier, dont la plupart des commentaires ont été supprimés pour plus de clarté, est indiqué dans l'EXEMPLE DE CODE 11-7.

Le fichier est situé par défaut dans `/etc/opt/SUNWspa`.

### Groupe `acl`

Le groupe `acl` contient les paramètres suivants :

- *noms de contextes*—Il s'agit d'une liste de noms de contextes séparés par une virgule.
- *accès*—Les valeurs possibles sont :
  - `lecture seule`
  - `lecture-écriture`

- *niveau de sécurité*—Les valeurs possibles sont :
  - noAuthNoPrivacy
  - authNoPrivacy
  - authPrivacy
- *utilisateurs*—Il s'agit d'une liste de noms d'utilisateur séparés par une virgule.

Dans l'exemple suivant, l'accès est accordé à l'utilisateur par défaut et toute requête émise par l'utilisateur par défaut dans le contexte public et nul avec une sécurité minimale authNoPrivacy est autorisée. Toutes les autres requêtes SNMP sont rejetées.

Ce fichier ne contient aucun groupe trap.

**EXEMPLE DE CODE 11-7** Exemple de fichier spapm.uacl

```
#ident "@(#)spama.uacl 1.4 01/29/03 SMI"
#
# Copyright 2003 Sun Microsystems, Inc. Tous droits réservés.
# Ce logiciel constitue la propriété exclusive de Sun Microsystems, Inc.
# Son utilisation est soumise à des conditions de licence.
#
# Modèle de fichier ACL
#
 acl = {
     {
         noms de contextes = public, nul
         accès = lecture-écriture
         niveau de sécurité = authNoPriv
         utilisateur = utilisateur par défaut
     }
 }
```

## Fichier spama.security

Le fichier spama.security indique les utilisateurs qui sont autorisés à accéder à l'agent maître, ainsi que le cryptage SNMPv3 et les clés d'authentification.

Il comprend un ensemble de lignes userEntry dotées du format suivant :

```
userEntry=<ID moteur>,<nom d'utilisateur>,<nom de sécurité>,<algorithme
d'authentification>, <clé d'authentification>, <algorithme de confidentialité>, <clé
de confidentialité>, <type de stockage>,<modèle>
```

**Précaution –** Ne modifiez que le paramètre de la ligne `userEntry` dans ce fichier.

Ces champs sont expliqués dans la TABLEAU 11-2.

**TABLEAU 11-2** Paramètres configurables par l'utilisateur dans `spama.security`

| Paramètre | Description |
|---|---|
| ID du moteur | ID du moteur SNMP à utiliser. Il peut être :<br>■ une chaîne hexadécimale ;<br>■ une chaîne textuelle représentant un `ID de moteur` sous la forme <`adresse`> :<`port`>:<`numéro IANA`> ;<br>■ la chaîne `localEngineID`, appropriée dans la plupart des cas. |
| nom d'utilisateur | Nom d'utilisateur auquel cette entrée s'applique |
| nom de sécurité | Nom de sécurité à mapper à ce nom d'utilisateur. Ils doivent être normalement identiques. |
| algorithme d'authentification | Algorithme d'authentification à utiliser. Celle-ci peut être :<br>• `usmHMACMD5AuthProtocol`<br>• `usmHMACSHHAuthProtocol`<br>• `usmNoAuthProtocol` |
| clé d'authentification | Clé à utiliser avec l'algorithme d'authentification. Celle-ci peut être :<br>• un mot de passe texte (huit caractères minimum) ;<br>• un code hexadécimal localisé. Par exemple, `0x0098768905AB67EFAA855A453B665B12.` |
| algorithme de confidentialité | Algorithme de confidentialité à utiliser. Celle-ci peut être :<br>• `usmDESPrivProtocol`<br>• `usmNoPrivProtocol` (paramètre par défaut si aucune valeur spécifiée) |
| clé de confidentialité | Clé à utiliser avec l'algorithme de confidentialité. Celle-ci peut être :<br>• un mot de passe texte (huit caractères minimum) ;<br>• un code hexadécimal localisé. Par exemple, `0x0098768905AB67EFAA855A453B665B12.` |
| type de stockage | La seule valeur acceptable est `3`, paramètre par défaut si vous ne spécifiez aucune valeur. |
| modèle | La valeur par défaut est `false` (il n'est pas nécessaire de modifier cette valeur) |

Un exemple du fichier, dont la plupart des commentaires ont été supprimés pour plus de clarté, est indiqué dans l'EXEMPLE DE CODE 11-8.

Le fichier est situé par défaut dans `/etc/opt/SUNWspa`.

Le fichier spama.security par défaut contient deux exemples d'utilisateurs que vous pouvez modifier et dont vous pouvez supprimer les commentaires pour définir vos propres utilisateurs. Le premier spécifie un utilisateur appelé `Utilisateur par défaut` doté des propriétés suivantes :

- authentification utilisant uniquement l'algorithme MD5 ;
- aucune confidentialité ;
- le mot de passe d'authentification « `mypassword` ».

Le second spécifie un utilisateur appelé `Utilisateur par défaut`, doté des propriétés suivantes :

- une authentification utilisant l'algorithme MD5 et le mot de passe d'authentification « `mypassword` » ;
- une confidentialité utilisant l'algorithme DES et le mot de passe de confidentialité « mypassword » ;
- une confidentialité utilisant l'algorithme DES

**EXEMPLE DE CODE 11-8** Exemple de fichier `spama-security`

```
#ident  "@(#)spama.security 1.7 01/29/03 SMI"
#
# Copyright 2002 Sun Microsystems, Inc. Tous droits réservés.
# Ce logiciel constitue la propriété exclusive de Sun Microsystems, Inc.
# Son utilisation est soumise à des conditions de licence.
#
# Modèle de fichier de sécurité

# localEngineBoots=0

# configuration de l'utilisateur par défaut. Authentification seulement.
# userEntry=localEngineID,defaultUser,,usmHMACMD5AuthProtocol,mypasswd

# configuration de l'utilisateur par défaut. Authentification et cryptage.
# userEntry=
localEngineID,defaultUser,null,usmHMACMD5AuthProtocol,mypasswd,usmDESPrivProt
ocol,mypasswd,3,
```

CHAPITRE **12**

# Configuration du logiciel

Ce chapitre décrit la configuration par défaut une fois l'installation terminée et explique comment modifier les fichiers décrit au Chapitre 11.

Il contient les sections suivantes :



## Configuration par défaut

Le logiciel est installé avec la configuration par défaut suivante :

- L'agent maître est désactivé (`START_MASTER_AGENT=no`).
- Le médiateur est activé (`START_MEDIATOR_AGENT=yes`).
- Le médiateur est configuré comme sous-agent de `snmpdx`.

---

**Remarque –** Pour des motifs de sécurité, vous devez configurer le fichier ACL de `snmpdx` de façon à restreindre l'accès pour exclure tous les systèmes autres que ceux qui surveillent l'agent.

---

## Contrôle d’accès

Pour permettre au médiateur d’utiliser la fonction Contrôle d’accès, configurez le fichier ACL du médiateur, comme indiqué dans le « Fichier `spapm.acl` », page 99.

Si vous utilisez `snmpdx` (configuration par défaut), modifiez `spapm_snmpdx.acl` pour configurer les autorisations d’accès et `spapm.acl` pour définir les destinataires des traps.

## Démarrage et arrêt du médiateur

Démarrez le médiateur à l’aide du script de démarrage standard snmpdx :

```
# /etc/init.d/init.snmpdx start
```

Arrêtez le médiateur à l’aide du script de médiateur :

```
# /etc/init.d/spama stop
```

# Configuration manuelle pour un accès direct

Étant donné que `snmpdx` prend uniquement en charge SNMPv1, vous pouvez configurez le port utilisé par le médiateur pour activer l’accès direct à SNMPv2c si vous souhaitez recourir aux opérations `get-bulk` sans utiliser l’agent maître.

Pour configurer manuellement le médiateur, effectuez les changements suivants dans `spama.conf`:

1. **Définissez** `SUB_AGENT=no`**.**
2. **Définissez** `SPAPM_REQ_PORT` **sur le numéro de port requis.**

   Les requêtes de SNMPv2c envoyées au médiateur doivent être transmises par ce port.

## Médiateur comme sous-agent d'un agent maître tiers

Pour configurer le médiateur comme sous-agent d'un agent maître tiers prenant en charge la spécification d'un numéro de port au moyen d'un paramètre de ligne de commande :

1. **Configurez le fichier ACL du médiateur pour autoriser l'accès depuis l'hôte local (voir « Fichier `spapm.acl` », page 99).**
2. **Configurez l'agent maître afin qu'il lance le médiateur avec l'invocation suivante, à l'aide d'un numéro de port approprié :**

```
port /etc/init.d/spama start <>
```

3. **Configurez l'agent maître afin qu'il envoie les requêtes aux sous-arbres d'IDO suivants :**
   - .iso.org.dod.internet.mgmt.mib-2.entityMIB
   - .iso.org.dod.internet.private.enterprises.sun.products.sunFire.sunPlatMIB

   ou par voie numérique :
   - .1.3.6.1.2.1.47
   - .1.3.6.1.4.1.42.2.70.101

# Le médiateur et l'agent maître SNMPv3

Pour activer le médiateur et l'agent maître, vous devez apporter au moins les modifications suivantes :

1. **Dans le fichier spama.conf :**
   a. **Définissez** `START_MASTER_AGENT=yes`.
   b. **Définissez** `SUB_AGENT=no`.
2. **Configurez le fichier ACL du médiateur afin qu'il autorise l'accès depuis l'hôte local, comme indiqué dans le « Fichier `spapm.acl` », page 99.**
3. **Configurez la commande snmpdx pour autoriser l'accès depuis l'hôte local, comme indiqué dans le « Fichier `spapm_snmpdx.acl` », page 101.**

4. **Configurez le fichier ACL du médiateur afin qu'il autorise l'accès aux gestionnaires souhaités depuis l'hôte local, comme indiqué dans le « Fichier `spapm.acl` », page 99.**

5. **Configurez les fichiers de sécurité pour définir les utilisateurs de SNMPv3, le contexte, ainsi que les niveaux d'authentification et de cryptage, comme indiqué dans les « Fichier `spama.acl` », page 103 et « Fichier `spama.security` », page 105.**

## Démarrage et arrêt des agents

Démarrez le médiateur et l'agent maître à l'aide du script du médiateur :

```
# /etc/init.d/spama start
```

Arrêtez le médiateur et l'agent maître à l'aide du script du médiateur :

```
# /etc/init.d/spama stop
```

## Envoi de traps SNMPv3

Pour configurer l'agent maître afin qu'il envoie des traps SNMPv3 depuis le médiateur, dans le fichier spama.conf (voir « `spama.conf` », page 86):

1. **Définissez** `SPAPM_TRAPS_ARE_V3=yes`**.**

2. **Vous pouvez également définir le** `SNMPV3_USER`**.**

---

**Remarque –** L'utilisateur du trap doit être configuré comme un utilisateur SNMPV3 dans `spama.uacl` et `spama.security`.

---

CHAPITRE 13

# Désinstallation du logiciel

Ce chapitre décrit comment désinstaller le logiciel.

Pour désinstaller SNMP, il suffit généralement d'utiliser la commande `pkgrm` pour supprimer les modules installés. Cette opération permet de supprimer tous les fichiers et les liens appropriés, et de réactiver `snmpdx`.

Les modifications de la configuration apportées automatiquement par le logiciel SNMP reviennent à leur état d'origine. Cependant, si vous modifiez des paramètres de fichier externes, tels que le fichier ACL snmpdx, vous devez les restaurer manuellement après avoir supprimé le logiciel SNMP.

---

**Remarque –** Les procédures décrites ci-dessous ne permettent pas de désinstaller le module API Java SNMP, `SUNWjsnmp`. Si vous souhaitez réinstaller la version Solaris de ce module, vous devez d'abord supprimer l'API Java SNMP.

---

# Modules de l’agent plate-forme et de l’agent cible

Pour supprimer les modules de l’agent plate-forme du serveur agent de la plate-forme, saisissez :

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpjo \
SUNWbgodr SUNWbgod SUNWbgcmr SUNWbgcm SUNWbgpc SUNWbgptk
```

**Précaution –** Supprimez les modules SUNWjdrt et SUNWjsnmp avec précaution car il s’agit de modules système pouvant être utilisés par d’autres produits.

Pour supprimer les modules de la plate-forme cible des lames Sun Fire B100s, saisissez :

```
# pkgrm SUNWbgpr SUNWbgcm SUNWbgpc SUNWbgptk
```

# Modules de l’agent domaine

Pour supprimer les modules de l’agent domaine des lames Sun Fire B100s, saisissez :

```
# pkgrm SUNWbgpmr SUNWbgpm SUNWjdrt SUNWjsnmp SUNWbgpji \
SUNWbgidr SUNWbgcmr SUNWngcm SUNWbgpc SUNWbgptk
```

**Précaution –** Supprimez les modules SUNWjdrt et SUNWjsnmp avec précaution car il s’agit de modules système pouvant être utilisés par d’autres produits.

CHAPITRE **14**

# Dépannage

Ce chapitre contient des informations vous permettant de résoudre les problèmes liés au système.

## *Problème*

- **L'agent SNMP ne fournit aucune réponse lorsque j'utilise la configuration par défaut (`snmpdx`).**

1. **Assurez-vous que le médiateur est exécuté en saisissant :**

```
# ps -ef | grep spa.snmp
    root 15789     1  1 13:44:01 pts/2    0:00 /usr/j2se/bin/java
-Dcom.sun.spa.snmp.LOG_LEVEL=INFO -Djdmk.security.file=//etc
```

Si vous obtenez la même réponse que celle indiquée ci-dessus, le processus du médiateur est exécuté.

Arrêtez le médiateur et redémarrez-le en saisissant :

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

2. **Assurez-vous que la version correcte de Java est installée en saisissant :**

```
# /usr/j2se/bin/java -version
java version « 1.4.1_03 »
Java(TM) 2 Runtime Environment, Édition standard (version 1.4.1_03-b04)
Java HotSpot(TM) Client VM (version 1.4.1_03-b04, mode mixte)
```

Ce dernier doit signaler la version 1.4 ou supérieure. Si ce n'est pas le cas, installez le JDK Java 1.4, comme décrit dans la rubrique « Environnement Java », page 66.

3. **Assurez-vous que la version correcte de** SUNWjsnmp **est installée en saisissant :**

```
# pkginfo -l SUNWjsnmp | grep VERSION
   VERSION:  5.0
```

Si la version 1.0 est indiquée, supprimer le module SUNWjsnmp et réinstallez la version dans l'archive SUNWspa.*.tar.Z (voir « API SNMP Java », page 67).

4. **Assurez-vous que le fichier spama.conf contient les entrées suivantes :**

START_MASTER_AGENT=no

START_MEDIATOR=yes

SUB_AGENT=yes

5. **Assurez-vous que le médiateur est correctement enregistré dans la commande** snmpdx **en saisissant :**

```
# cat /var/snmp/snmpdx.st
spapm spapm 2516 34050
snmpd snmpd 2567 34053
```

L'entrée spapm ci-dessus indique que le médiateur est enregistré comme un sous-agent de snmpdx.

6. **Assurez-vous que** /etc/snmp/conf/spapm.reg **et** /etc/snmp/conf/spapm.rsrc **ne sont pas endommagés.**

Arrêtez le médiateur et redémarrez-le en saisissant :

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

7. **Assurez-vous que les autorisations sont correctement définies dans les fichiers ACL.**
   - spapm_snmpdx.acl définit l'accès au gestionnaire SNMP actuellement utilisé.
   - spapm.acl définit l'accès pour l'hôte local.

Pour de plus amples informations, voir Chapitre 11.

*Problème*

- **Il n'existe aucune instrumentation pour le disque dur (HHD) ou d'adresse MAC Ethernet lorsque j'utilise l'agent plate-forme.**

  Ces informations sont uniquement disponibles lorsque le système d'exploitation est exécuté sur la lame Sun Fire B100s cible.

1. **Assurez-vous que la lame Sun Fire B100s est amorcée.**
2. **Assurez-vous que l'instrumentation cible est exécutée sur la lame Sun Fire B100s en saisissant :**

```
# netstat -an | grep 1099
      *.1099        *.*         0      0 24576      0 LISTEN
```

Si aucun port n'est à l'écoute, cela signifie que l'instrumentation cible n'est pas exécutée.

Démarrez l'instrumentation sur la lame Sun Fire B100s en saisissant :

```
# /etc/init.d/spama start
```

3. **Assurez-vous que la version correcte de Java est installée en saisissant :**

```
# /usr/j2se/bin/java -version
java version « 1.4.1_03 »
Java(TM) 2 Runtime Environment, Édition standard (version 1.4.1_03-b04)
Java HotSpot(TM) Client VM (version 1.4.1_03-b04, mode mixte)
```

Ce dernier doit signaler la version 1.4 ou supérieure. Un version incorrecte peut causer le démarrage de l'instrumentation pour une courte durée et sa panne.

Si ce n'est pas le cas, installez le JDK Java 1.4, comme décrit dans la rubrique « Environnement Java », page 66.

*Problème*

- **L'agent est accessible, mais il n'existe aucune instrumentation pour les plates-formes surveillées**

1. **Assurez-vous que le démon de détection est exécuté en saisissant :**

```
# netstat -a | grep mismi
      *.mismi              *.*                0      0 24576      0 LISTEN
      *.mismi                              *.*  0
0 24576      0 LISTEN
```

La sortie ci-dessus indique que le démon de détection est à l'écoute des requêtes provenant des plates-formes gérées.

**a. Assurez-vous que** `/etc/services` **inclut l'entrée suivante :**

```
mismi           8265/tcp                # MISMI Discovery
```

**b. Assurez-vous que** `/etc/inetd.conf` **inclut l'entrée suivante :**

```
# MISMIDISCOVERY - mismiDiscovery daemon
mismi   stream  tcp6    nowait  root    /opt/SUNWspa/bin/mismiDiscovery
mismiDiscovery
```

**c. Assurez-vous que** `/etc/inetd.conf` **est un lien symbolique avec** `/etc/inet/inetd.conf` **en saisissant :**

```
# ls -l /etc/inetd.conf
lrwxrwxrwx   1 root     root          17 Jan  7 17:08 /etc/inetd.conf ->
./inet/inetd.conf
```

Si le lien n'existe pas, la mise à jour du fichier échoue lors de l'installation du module `SUNWbgodr`.

Corrigez la configuration `inetd` et redémarrez en saisissant :

```
# pkill -1 inetd
```

2. **Assurez-vous que la plate-forme a été détectée en saisissant :**

```
# # netstat -a | grep mismi
      *.mismi              *.*                0      0 24576      0 LISTEN
blade-174-119.36780  hornet-sc.mismi       8192      0 24820      0 ESTABLISHED
      *.mismi                           *.*  0
0 24576      0 LISTEN
```

La sortie indique que le démon de détection est à l'écoute et qu'une connexion avec le contrôleur du système de la plate-forme (appelée <hornet-sc>) est établie.

Si aucune connexion n'est établie, vérifiez la configuration du contrôleur du système, comme indiquée au chapitre « Configuration du contrôleur du système », page 79.

### *Problème*

- **Le délai d'exécution des requêtes** get **et** set **de SNMPv3 est dépassé.**
  - **Cause probable**

L'identificateur du moteur local ou le nombre de localEngineBoots du fichier spama.security ont peut-être été modifiés ou supprimés.

  - **Vérifiez**

Il n'est pas évident de savoir si le fichier a été ou non modifié.

  - **Résolution**

Redémarrez l'agent, comme indiqué ci-dessous, puis l'application de gestion pour les synchroniser de nouveau.

```
# /etc/init.d/spama stop
# /etc/init.d/spama start
```

*Problème*

- **Le délai d'exécution des requêtes `get` et `set` de SNMP est dépassé.**

  - **Cause probable**

  Sur un système extrêmement chargé, il est possible que le délai d'exécution des requêtes de l'agent maître `snmpdx` expire dans le médiateur SNMP. Le délai d'expiration est actuellement défini à 2s (2000000 s).

  - **Vérification**

  Il n'est pas évident de savoir si le délai d'exécution signalé par une application de gestion a effectivement expiré entre `snmpdx` et le médiateur SNMP, ou entre l'application de gestion et `snmpdx`.

  - **Correction**

  Vous pouvez augmenter le délai d'expiration en modifiant sa propriété dans le fichier `/etc/snmp/conf/spapm.reg`. Si vous modifiez le fichier, redémarrez le médiateur en saisissant :

```
# /etc/init.d/spama stop
# /etc/init.d/init.snmpdx start
```

ANNEXE **A**

# Installation de J2RE 1.4 pour une utilisation conjointe avec J2SE 1.3.1

Cette annexe décrit comment installer l'environnement d'exécution Java 2 (J2RE) Édition standard 1.4 pour une utilisation conjointe avec J2SE 1.3.1 sur le serveur agent de la plate-forme et les domaines B100s, et comment modifier les scripts de démarrage pour localiser l'installation.

Elle contient les sections suivantes :



## Installation de J2RE 1.4

Pour installer J2RE 1.4 pour une utilisation conjointe avec J2SE 1.3.1, abordée dans la rubrique « Environnement Java », page 66, observez la procédure ci-dessous.

J2RE 1.4 est disponible comme fichier binaire auto-extractible à partir de :

`http://java.sun.com/j2se/1.4/download.html`

Suivez les instructions ci-dessous pour installer J2RE. Des informations détaillées sur le téléchargement du fichier sont disponibles sur le site Web indiqué ci-dessus.

---

**Remarque –** Ce produit requiert uniquement une prise en charge de 32 bits. Il n'est donc pas nécessaire d'installer le complément de 64 bits pour l'environnement J2RE.

---

Dans les étapes suivantes, remplacez le numéro de version de la mise à jour approprié de J2RE contenu dans les étapes suivantes par `<ver>`.

Par exemple, si vous téléchargez la mise à jour `1.4.0_01`, la commande suivante :

```
# chmod +x j2re-1_4_<version>-solaris-sparc.sh
```

devient :

```
chmod +x j2re-1_4_0_01-solaris-sparc.sh
```

1. **Téléchargez et vérifiez la taille du fichier**

   Le fichier requis est :

   `j2re-1_4_<ver>-solaris-sparc.sh`

   Avant de télécharger un fichier, notez sa taille, mentionnée dans la page de téléchargement. Une fois le téléchargement terminé, vérifiez que vous avez téléchargé le fichier logiciel complet et non endommagé.

   Assurez-vous d'avoir téléchargé le fichier à un emplacement accessible à la racine (par exemple, dans `/tmp`).

2. **Allez à la racine en exécutant `su` et en saisissant le mot de passe super-utilisateur.**

3. **Assurez-vous que les autorisations d'exécution sont définies dans le fichier binaire auto-extractible.**

```
# chmod +x j2re-1_4_<ver>-solaris-sparc.sh
```

4. **Modifiez le répertoire où doivent être installés les fichiers.**

```
# cd /usr
```

5. **Exécutez le fichier binaire auto-extractible.**

   Un répertoire appelé `/usr/j2re1.4.<ver>` contenant l'environnement J2RE est créé.

6. **Assurez-vous que l'environnement J2RE est correctement installé.**

```
# /usr/j2re1.4.1_01/bin/java -version
java version « 1.4.1_01 »
Java(TM) 2 Runtime Environment, Édition standard (version
1.4.1_01-b01)
Java HotSpot(TM) Client VM (version 1.4.1_01-b01, mode mixte)
```

   Ce dernier doit signaler la version `1.4` ou supérieure. Cet exemple indique la version `1.4.1_01`.

7. **Supprimez le fichier binaire auto-extractible.**
8. **Quittez le shell de la racine.**

# Modification des scripts de démarrage

Cette section décrit comment modifier les scripts de démarrage lors de l'installation de J2RE 1.4.

Lisez cette section conjointement avec le Chapitre 10.

## Surveillance du domaine matérielle

Ces étapes concernent l'Step 4 du chapitre « Installation du logiciel pour une surveillance de domaine », page 73.

1. **Sur chaque lame B100s surveillée, modifiez chacun des scripts de démarrage suivants :**
   - `/etc/init.d/spaibdm`
   - `/etc/init.d/spapom`

   en remplaçant la ligne

   `JAVA=/usr/j2se/bin/java`

   par

   `JAVA=/usr/j2re1.4.<ver>/bin/java`
2. **Sur chaque lame B100s surveillée, modifiez le script de démarrage suivant :**
   - `/etc/init.d/spama`

   en remplaçant la ligne

   `JAVA_JAVA=/usr/j2se/bin/java`

   par

   `JAVA_JAVA=/usr/j2re1.4.<ver>/bin.java`

# Surveillance de la plate-forme matérielle

Les étapes 1 et 2 concernent l'Step 4 du chapitre « Pour installer le logiciel avec l'instrumentation de la cible », page 75, et Step 4 du chapitre « Pour installer le logiciel sans instrumentation de la cible », page 77.

L'étape 3 ne concerne que l'Step 11 « Pour installer le logiciel sans instrumentation de la cible », page 77.

1. **Sur le serveur agent de la plate-forme, modifiez chacun des scripts de démarrage suivants :**
   - `/etc/init.d/spapom`

   en remplaçant la ligne

   `JAVA=/usr/j2se/bin/java`

   par

   `JAVA=/usr/j2re1.4.<ver>/bin/java`

2. **Sur le serveur agent de la plate-forme, modifiez le script de démarrage suivant :**
   `/etc/init.d/spama`

   en remplaçant la ligne

   `JAVA_JAVA=/usr/j2se/bin/java`

   par

   `JAVA_JAVA=/usr/j2re1.4.<ver>/bin.java`

3. **Sur chaque lame B100s surveillée (cible), modifiez le script de démarrage suivant :**
   - `/etc/init.d/spardp`

   en remplaçant la ligne

   `JAVA=/usr/j2se/bin/java`

   par

   `JAVA=/usr/j2re1.4.<ver>/bin/java`

# Index

## D



## E



## F

## N



## O



## P



## R



## S

## T



## U



## V