Tivoli Application Dependency Discovery Manager
Version 7.3

Guide du développeur SDK

Tivoli Application Dependency Discovery Manager
Version 7.3

Guide du développeur SDK

Remarque

Avant d'utiliser le présent document et le produit associé, prenez connaissance des informations figurant à la section «Remarques», à la page 191.

Notice d'édition

La présente édition s'applique à la version 7.3 d'IBM Tivoli Application Dependency Discovery Manager (numéro de produit 5724-N55) et à toutes les éditions et modifications ultérieures jusqu'à indication contraire dans les nouvelles éditions.

Table des matières

Tableaux

Avis aux lecteurs canadiens

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

Illustrations

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

Terminologie

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

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

Claviers

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

OS/2 et Windows - Paramètres canadiens

Au Canada, on utilise :

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

Nomenclature

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

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

Brevets

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

Assistance téléphonique

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

A propos de la présente documentation

Ce document PDF est la version imprimable des informations fournies par le centre de documentation.

Conventions utilisées dans ce centre de documentation

Dans la documentation d'IBM® Tivoli Application Dependency Discovery Manager (TADDM), certaines conventions sont utilisées. Elles font référence aux variables et chemins d'accès liés au système d'exploitation, au répertoire `COLLATION_HOME`, ainsi qu'à l'emplacement du fichier `collation.properties` mentionné dans la documentation de TADDM, y compris dans les messages.

Variables et chemins dépendant du système d'exploitation

Dans ce centre de documentation, les conventions UNIX sont utilisées pour spécifier des variables d'environnement et pour la notation des répertoires.

Si vous utilisez une ligne de commande Windows, remplacez $*variable* par %*variable*% pour les variables d'environnement, et remplacez toutes les barres obliques (/) par des barres obliques inverses (\) dans les chemins d'accès des répertoires.

Si vous utilisez l'interpréteur de commandes bash dans un système Windows, utilisez les conventions UNIX.

Répertoire `COLLATION_HOME`

Le répertoire principal TADDM est également appelé répertoire `COLLATION_HOME`.

Sur les systèmes d'exploitation tels que AIX ou Linux, l'emplacement par défaut pour l'installation de TADDM est le répertoire `/opt/IBM/taddm`. Par conséquent, l'emplacement du répertoire `$COLLATION_HOME` est `/opt/IBM/taddm/dist`.

Sur les systèmes d'exploitation Windows, l'emplacement d'installation par défaut de TADDM est le répertoire `c:\IBM\taddm`. Dans ce cas, l'emplacement du répertoire `%COLLATION_HOME%` est `c:\IBM\taddm\dist`.

Emplacement du fichier `collation.properties`

Le fichier `collation.properties` renferme les propriétés du serveur TADDM et inclut des commentaires sur chacune d'elles. Il se trouve dans le répertoire `$COLLATION_HOME/etc`.

Termes et définitions

Reportez-vous à la liste de termes et définitions suivante pour en savoir plus sur les principaux concepts d'IBM Tivoli Application Dependency Discovery Manager (TADDM).

collection d'accès
Collection permettant de contrôler l'accès aux éléments de configuration et

les droits de modification des éléments de configuration. Vous ne pouvez créer des collections d'accès que si la sécurité au niveau des données est activée.

reconnaissance asynchrone
Dans TADDM, l'exécution d'un script de reconnaissance sur un système cible permettant de reconnaître des systèmes auxquels le serveur TADDM n'a pas directement accès. Cette reconnaissance s'effectuant manuellement et indépendamment d'une reconnaissance authentifiée, elle est dite «asynchrone».

application métier
Collection de composants offrant une fonctionnalité métier que vous pouvez utiliser en interne, en externe ou avec d'autres applications métier.

EC Voir *élément de configuration*.

collection
Dans TADDM, groupe d'éléments de configuration.

élément de configuration (EC)
Un composant de l'infrastructure informatique sous le contrôle de la gestion des configurations et donc soumis à un contrôle formel des modifications. Chaque EC dans la base de données TADDM possède un objet persistant et un historique des changements qui lui sont associés. Exemples d'EC : système d'exploitation, interface L2, taille du pool de mémoire tampon de base de données.

reconnaissance authentifiée
L'analyse du détecteur TADDM permettant de reconnaître des informations détaillées sur les éléments suivants :

- Chaque système d'exploitation dans l'environnement d'exécution. Cette analyse est également appelée reconnaissance de niveau 2 et requiert les droits d'accès au système d'exploitation.
- L'infrastructure d'application, les composants logiciels déployés, les serveurs physiques, les périphériques réseau, les systèmes virtuels et les données hôtes utilisés dans un environnement d'exécution. Cette analyse est également appelée reconnaissance de niveau 3 et requiert les droits d'accès au système d'exploitation et à l'application.

reconnaissance non authentifiée
L'analyse du détecteur TADDM permettant de découvrir des informations de base relatives aux systèmes informatiques actifs dans l'environnement d'exécution. Cette analyse est également appelée reconnaissance de niveau 1 et ne requiert aucun droit d'accès.

Portail de gestion de données
L'interface utilisateur Web de TADDM permettant d'afficher et de manipuler les données d'une base de données TADDM. Elle s'applique à un déploiement de serveur de domaine, à un déploiement de serveur de synchronisation et à chaque serveur de stockage dans un déploiement de serveur de diffusion en continu. L'interface utilisateur est très similaire dans tous les déploiements, bien qu'elle comporte quelques fonctions supplémentaires permettant d'ajouter et de synchroniser des domaines dans le déploiement de serveur de synchronisation.

unité d'exécution de tâche de reconnaissance
Dans TADDM, unité d'exécution qui exécute des détecteurs.

Console de gestion de reconnaissance
L'interface utilisateur client TADDM permettant de gérer les reconnaissances. Cette console est également appelée console produit. Elle s'applique au déploiement d'un serveur de domaine et au déploiement de serveurs de reconnaissance dans un déploiement de serveurs de diffusion en continu. La fonction de la console est la même dans ces deux déploiements.

serveur de reconnaissance
Un serveur TADDM qui exécute des détecteurs dans un déploiement de serveurs de diffusion en continu mais qui ne possède pas sa propre base de données.

domaine
Dans TADDM, un sous-ensemble logique de l'infrastructure d'une société ou d'une autre organisation. Les domaines peuvent représenter des limites organisationnelles, fonctionnelles ou géographiques.

serveur de domaine
Un serveur TADDM exécutant des détecteurs dans un déploiement de serveur de domaine et possédant sa propre base de données.

déploiement de serveur de domaine
Un déploiement TADDM possédant un serveur de domaine. Un déploiement de serveur de domaine peut faire partie d'un déploiement de serveur de synchronisation.

Dans un déploiement de serveur de domaine, la propriété suivante du serveur TADDM doit être définie sur la valeur suivante :

```
com.collation.cmdbmode=domain
```

lancement en contexte
Le concept consistant à passer de façon homogène d'une interface utilisateur de produit Tivoli (soit sur une console différente, soit sur la même console ou interface de portail) avec une identification unique et avec l'interface utilisateur cible en position sur l'emplacement correct pour que les utilisateurs poursuivent leur tâche.

reconnaissance de niveau 1
L'analyse du détecteur TADDM permettant de découvrir des informations de base relatives aux systèmes informatiques actifs dans l'environnement d'exécution. Cette analyse est également intitulée reconnaissance sans autorisation d'accès car elle ne requiert aucune autorisation d'accès. Elle utilise le détecteur Stack Scan et le détecteur de portée IBM® Tivoli® Monitoring. La reconnaissance de niveau 1 est très superficielle. Elle collecte uniquement le nom hôte, le nom du système d'exploitation, l'adresse IP, le nom de domaine complet et l'adresse MAC (Media Access Control) de chaque interface reconnue. De plus, la reconnaissance des adresses MAC se limite aux systèmes Linux on System z® et Windows. La reconnaissance de niveau 1 ne permet pas de reconnaître les sous-réseaux. Pour chaque interface IP reconnue qui n'appartient pas à un sous-réseau existant reconnu lors d'une reconnaissance de niveau 2 ou 3, de nouveaux sous-réseaux sont créés en fonction de la valeur de la propriété `com.collation.IpNetworkAssignmentAgent.defaultNetmask` du fichier `collation.properties`.

reconnaissance de niveau 2
L'analyse du détecteur TADDM permettant de découvrir des informations détaillées sur chaque système d'exploitation de l'environnement d'exécution. Cette analyse est également appelée reconnaissance avec

autorisation d'accès car elle requiert les autorisations d'accès au système d'exploitation. La reconnaissance de niveau 2 collecte les noms des applications, les noms des systèmes d'exploitation et les numéros de port associés à chaque application en cours d'exécution. Si une application a établi une connexion TCP/IP avec une autre application, ces informations sont capturées en tant que dépendance.

reconnaissance de niveau 3
L'analyse du détecteur TADDM reconnaît des informations détaillées sur l'infrastructure de l'application, les composants logiciels déployés, les serveurs physiques, les unités réseau, les systèmes virtuels et les données hôte utilisées dans l'environnement d'exécution. Cette analyse est également connue en tant que reconnaissance avec autorisations d'accès car elle requiert les autorisations d'accès au système d'exploitation et à l'application.

location multiple
Dans TADDM, l'utilisation par un fournisseur de services ou un vendeur informatique d'une installation TADDM pour découvrir plusieurs environnements clients. De plus, le fournisseur de services ou le vendeur informatique peut voir les données provenant de tous les environnements clients, mais au sein de chaque environnement client, seules les données spécifiques à un client peuvent être affichées dans l'interface utilisateur ou consultées dans les rapports inhérents à cet environnement client.

console produit
Voir *console de gestion de reconnaissance*.

reconnaissance basée sur un script
Dans TADDM, l'utilisation dans une reconnaissance authentifiée de scripts de détecteur identiques à ceux fournis par les détecteurs dans le support de reconnaissance asynchrone.

ES Voir *équivalent serveur*.

équivalent serveur (ES)
Une unité représentative de l'infrastructure informatique, définie comme un système informatique (avec des configurations standard, des systèmes d'exploitation, des interfaces réseau et des interfaces de stockage) avec un logiciel serveur installé (base de données, serveur Web ou serveur d'applications, par exemple). Le concept d'équivalent serveur inclut aussi le réseau, l'archivage et les autres sous-systèmes fournissant des services pour le fonctionnement optimal du serveur. Un serveur équivalent dépend du système d'exploitation :

Système d'exploitation	Nombre approximatif d'EC
Windows	500
AIX	1000
Linux	1000
HP-UX	500
Périphériques réseau	1000

serveur de stockage
Un serveur TADDM qui traite les données reçues des serveurs de reconnaissance et les enregistre dans la base de données TADDM. Le serveur de stockage principal coordonne les serveurs de reconnaissance ainsi que tous les autres serveurs de stockage et fait office de serveur de

stockage. Tous les serveurs de stockage qui ne sont pas des serveurs principaux sont appelés serveurs de stockage secondaires.

déploiement de serveurs de diffusion en continu
Un déploiement TADDM avec un serveur de stockage principal et au moins un serveur de reconnaissance. Ce type de déploiement peut également inclure un ou plusieurs serveurs de stockage secondaires en option. Le serveur de stockage principal et les serveurs de stockage secondaires partagent une même base de données. Les serveurs de reconnaissance ne comportent aucune base de données.

Dans ce type de déploiement, les données de reconnaissance affluent en parallèle de plusieurs serveurs de reconnaissance pour converger vers la base de données TADDM.

Dans un déploiement de serveurs de diffusion en continu, la propriété du serveur TADDM doit être définie sur l'une des valeurs suivantes :

- `com.collation.taddm.mode=DiscoveryServer`
- `com.collation.taddm.mode=StorageServer`

Pour tous les serveurs, à l'exception du serveur de stockage principal, les propriétés suivantes (pour le nom d'hôte et le numéro de port du serveur de stockage principal) doivent également être définies :

- `com.collation.PrimaryStorageServer.host`
- `com.collation.PrimaryStorageServer.port`

Si la propriété com.collation.taddm.mode est définie, la propriété com.collation.cmdbmode ne doit pas être définie ou elle doit être placée en commentaire.

serveur de synchronisation
Un serveur TADDM qui synchronise les données de reconnaissance à partir de tous les serveurs de domaine de l'entreprise et qui comporte sa propre base de données. Ce serveur ne reconnaît pas directement les données.

déploiement de serveur de synchronisation
Un déploiement TADDM avec un serveur de synchronisation et deux ou plusieurs déploiements de serveur de domaine comportant chacun sa propre base de données locale.

Dans ce type de déploiement, le serveur de synchronisation copie les données de reconnaissance de plusieurs serveurs de domaine, un domaine à la fois, au cours d'un processus de synchronisation par lots.

Dans un déploiement de serveur de synchronisation, la propriété suivante du serveur TADDM doit être définie sur l'une des valeurs suivantes :

`com.collation.cmdbmode=enterprise`

Ce type de déploiement est obsolète. Par conséquent, dans un nouveau déploiement TADDM, dans lequel plusieurs serveurs sont requis, utilisez le déploiement de serveurs de diffusion en continu. Vous pouvez convertir un serveur de synchronisation en serveur de stockage principal d'un déploiement de serveurs de diffusion en continu.

base de données TADDM
Dans TADDM, la base de données dans laquelle les données de configuration, les dépendances et l'historique des changements sont enregistrés.

Chaque serveur TADDM, à l'exception des serveurs de reconnaissance et des serveurs de stockage secondaires, possède sa propre base de données. Les serveurs de reconnaissance ne comportent aucune base de données. Les serveurs de stockage partagent la base de données du serveur de stockage principal.

serveur TADDM
Une dénomination générique pouvant représenter l'une des dénominations suivantes :

- Serveur de domaine dans un déploiement de serveur de domaine
- Serveur de synchronisation dans un déploiement de serveur de synchronisation
- Serveur de reconnaissance dans un déploiement de serveur de reconnaissance
- Serveur de stockage (y compris le serveur de stockage principal) dans un déploiement de serveurs de diffusion en continu

système cible
Dans le processus de reconnaissance TADDM, le système devant être reconnu.

reconnaissance d'utilisation
L'analyse du détecteur TADDM reconnaît les informations d'utilisation du système hôte. La reconnaissance d'utilisation requiert les autorisations d'accès au système d'exploitation.

Guide de développement de logiciels du développeur

Présentation du logiciel SDK

Ce chapitre présente le logiciel SDK TADDM (IBM Tivoli Application Dependency Discovery Manager) et offre un rapide aperçu du Common Data Model de TADDM.

Le Guide du développeur de logiciel SDK apporte une visibilité précise des applications de gestion grâce à des mappes d'application mettant en évidence la relation entre l'application et son infrastructure de support. Les mappes d'application étendues incluent les composants d'infrastructure qui composent l'application, leurs configurations détaillées ainsi que les dépendances et interrelations d'exécution.

TADDM stocke les données de topologie en interne selon une hiérarchie d'objet Java™ connue sous le nom de Common Data Model (CDM).

Présentation du kit de l'éditeur de logiciels (logiciel SDK)

Ce manuel du logiciel SDK utilise l'architecture ouverte et évolutive de TADDM et vous offre un moyen de réutiliser rapidement et de manière efficace l'ensemble des mappes d'applications à travers diverses solutions de gestion d'applications.

Ce manuel offre un accès complet aux mappes d'applications et au processus de reconnaissance, avec lesquels vous pouvez :

- Protéger les investissements en implémentation en utilisant un logiciel SDK normalisé, ouvert et établi commercialement
- Garantir la réussite des initiatives de gestion informatique en partageant et réutilisant de manière rentable les mappes d'application TADDM à travers les applications de gestion
- Améliorer la précision des solutions de gestion en intégrant des mappes d'application exactes et en temps réel
- Utiliser les intégrations et les adaptateurs TADDM pour des déploiements efficaces

Le logiciel SDK TADDM offre un ensemble d'interfaces de programme d'application documenté :

- Interface de programme d'application Java
- Interface de programme d'application du protocole SOAP
- Interface de programme d'application REST
- Interface de programme d'application de l'interface de ligne de commande

Ces interfaces de programme d'application offrent un accès complet aux mappes d'applications TADDM, y compris aux applications reconnues, à leurs composants, configurations et dépendances. Les interfaces du programme d'application offrent également un contrôle total du processus de reconnaissance TADDM et de son cycle de vie (démarrage, arrêt et gestion des reconnaissances).

Présentation du Common Data Model

TADDM stocke les données de topologie en interne selon une hiérarchie d'objet Java connue sous le nom de Common Data Model.

Le Common Data Model (CDM), qui est conservé dans une base de données relationnelle, se compose d'objets de modèle qui représentent des éléments reconnus dans l'environnement de l'entreprise. Le modèle de données contient des objets reconnus de chaque type d'élément, tels que des systèmes informatiques ou des applications, avec les caractéristiques correspondantes représentées comme des objets contenus, tels que des systèmes d'exploitation ou des valeurs de configuration.

Vous pouvez accéder au modèle à l'aide de l'interface de programme d'application IBM TADDM, avec toutes les données des caractéristiques affichées dans le portail de gestion de données accessible depuis cette interface. Le logiciel SDK représente les données au format XML avec un schéma XML publié. La plupart des objets contenus sont imbriqués dans le document et les objets qui sont référencés plusieurs fois sont créés en double au sein du document. Le document XML obtenu est légèrement plus volumineux que les données d'origine, mais la recherche est simplifiée grâce à des outils tels que XQuery ou Xpath.

Concepts associés:

«Simplified Model», à la page 19
Au vu des problèmes causés par le Common Data Model, un nouveau Simplified Data Model pour le stockage de données a été introduit dans la version 7.3 de TADDM. Les classes sont les seuls éléments conservés de l'ancien modèle.

Installation et configuration du kit de l'éditeur de logiciels

Ce chapitre décrit la configuration système requise pour utiliser le kit de développement de logiciels TADDM (IBM Tivoli Application Dependency Discovery Manager) et explique comment l'installer et le configurer sur votre ordinateur.

Configuration requise

Cette section décrit la configuration requise pour l'utilisation du logiciel SDK TADDM.

Le tableau 1 établit une liste des éléments système et décrit les détails respectifs de la condition.

Tableau 1. Configuration requise pour SDK TADDM

Elément	**Caractéristiques**
Système d'exploitation	Tout système d'exploitation qui prend en charge l'environnement d'exécution obligatoire Java (JRE)
Mémoire	2 Go
Processeurs	1
Vitesse processeur	1 GHz
Espace disque	200 Mo (y compris la machine virtuelle Java)

Tableau 1. Configuration requise pour SDK TADDM (suite)

Elément	Caractéristiques
Configurations logicielles supplémentaires	Si vous exécutez le logiciel SDK sur le même ordinateur que le serveur TADDM, utilisez IBM Java SDK version 7.0 fourni avec le serveur TADDM. IBM Java SDK est situé dans le répertoire `$COLLATION_HOME/external`. Si vous installez le logiciel SDK sur un autre ordinateur que le serveur TADDM, la version 7.0 de Java SDK est nécessaire. Si le client ne se trouve pas sur la même machine que le serveur, les niveaux Java SDK doivent correspondre. Par exemple, n'essayez pas d'exécuter un client d'une version 5.0 de Java SDK avec un serveur version 7.0.

Installation du logiciel SDK TADDM

Ce chapitre décrit l'installation du logiciel SDK TADDM sur votre ordinateur.

Vous pouvez utiliser le logiciel SDK dans l'un des modes suivants :

- Mode imbriqué : le logiciel SDK est installé au moment de l'installation de TADDM sur votre système. Pour plus d'informations, consultez la rubrique sur le mode imbriqué.
- Mode autonome : ce mode permet d'installer le logiciel SDK sur des systèmes autonomes. Pour plus d'informations, consultez la rubrique sur le mode autonome.

Sur les systèmes multiutilisateurs tels que Linux et AIX, si plusieurs personnes utilisent le logiciel SDK, les droits d'accès aux fichiers journaux entrent en conflit. Pour éviter ce problème, vous pouvez installer le logiciel SDK dans votre répertoire de base.

Mode imbriqué

Si le serveur TADDM est déjà installé sur votre ordinateur, le logiciel SDK est disponible comme composant de la distribution dans le répertoire `$COLLATION_HOME/sdk`, comme indiqué dans le tableau ci-dessous.

Tableau 2. Arborescence des répertoires du mode imbriqué

Répertoire	Contenu
dist/	Répertoire racine de TADDM, également connu comme le répertoire COLLATION_HOME. Sur les systèmes d'exploitation tels que AIX ou Linux, l'emplacement par défaut pour l'installation de TADDM est le répertoire /opt/IBM/taddm. Par conséquent, l'emplacement du répertoire $COLLATION_HOME est /opt/IBM/taddm/dist. Sur les systèmes d'exploitation Windows, l'emplacement d'installation par défaut de TADDM est le répertoire c:\IBM\taddm. Dans ce cas, l'emplacement du répertoire %COLLATION_HOME% est c:\IBM\taddm\dist.
bin/	
deploy/	
etc/	
external/	
lib/	
log/	
dist/sdk/	Contient le logiciel SDK TADDM. Pour plus d'informations, voir le tableau sur la structure des répertoires du mode autonome.

Mode autonome

Pour installer le logiciel SDK TADDM séparément, accédez au répertoire $COLLATION_HOME/SDK et extrayez le fichier sdk.zip vers n'importe quel répertoire de votre système. La structure des répertoires du logiciel SDK est présentée dans le tableau ci-dessous :

Tableau 3. Arborescence des répertoires du mode autonome

Répertoire	Contenu
adaptor	Contient l'adaptateur de bibliothèque de reconnaissance TADDM 1.0.
bin	Contient des scripts de shell et des fichiers de traitement par lots utiles
dla	Contient l'outil de certification IdML de la bibliothèque de reconnaissance IBM
doc	Contient des fichiers PDF et d'autres fichiers de documentation en anglais
etc	Propriétés de configuration
examples	Répertoire des échantillons
lib	Bibliothèques d'exécution du serveur et du client
log	Journaux de la phase d'exécution
schema	Schéma XML

Configuration du logiciel SDK TADDM

Vous pouvez configurer le logiciel SDK TADDM en spécifiant des valeurs pour les variables d'environnement. Une option vous permet également de configurer l'exécution du logiciel SDK en spécifiant des valeurs pour les paramètres de configuration.

Définition des variables d'environnement

Avant de commencer

Vous devez définir des variables d'environnement avant d'utiliser l'interface de ligne de commande et le kit de l'éditeur de logiciels, ou avant d'exécuter les exemples fournis.

Procédure

Pour définir les variables d'environnement, procédez comme suit :

Définissez la variable d'environnement JAVA_HOME sur le répertoire pour l'environnement d'exécution Java.
Si JAVA_HOME n'est pas défini, le script exécute le premier fichier exécutable Java trouvé sur le chemin d'exécution.

Définition des propriétés de configuration

Les paramètres de configuration se trouvent dans le fichier `$COLLATION_HOME/sdk/etc/collation.properties`. Le tableau 4 décrit les paramètres de configuration que vous pouvez spécifier :

Tableau 4. Propriétés de configuration

Paramètre	Caractéristiques
`com.collation.version`	Version de l'interface de programme d'application
`com.collation.LogFile`	Emplacement de consignation des messages côté client. Le répertoire doit exister. Le fichier est créé le cas échéant. Le fichier fourni par défaut est `../log/api-client.log`. Si cette propriété n'est pas spécifiée, la consignation s'effectue par défaut dans stdout.
`com.collation.log.level`	Niveau de consignation, parmi les valeurs suivantes : • INFO—Default • ERROR • DEBUG
`com.collation.log.filesize`	Taille du fichier journal. La valeur par défaut est 20 Mo.
`com.collation.log.filecount`	Nombre de remise à zéro. La valeur par défaut est 3.

Tableau 4. Propriétés de configuration (suite)

Paramètre	Caractéristiques
com.ibm.cdb.service.registry.public.port	Port par défaut pour le registre RMI des services TADDM publics. L'interface de programme d'application du logiciel SDK représente un des services TADDM publics. Cette valeur doit être identique à celle du paramètre du serveur TADDM. La valeur par défaut est 9433. Si l'interface de programme d'application se connecte à plusieurs serveurs TADDM, vous devez configurer l'ensemble des serveurs pour qu'ils utilisent le même port ou spécifier le port lors de la connexion.

Le paramètre de propriété com.ibm.cdb.service.registry.public.port doit correspondre au paramètre du serveur TADDM. Sinon, le logiciel SDK TADDM ne fonctionne pas. Cette condition est requise à la fois pour les modes imbriqué et autonome.

Vérification de l'installation du logiciel SDK

Avant de commencer

Vous pouvez vérifier si vous avez correctement installé et configuré le logiciel SDK TADDM.

Procédure

Pour vérifier que l'installation s'est correctement déroulée, procédez comme suit :

1. Indiquez le répertoire binaire SDK en lançant une commande semblable à celle qui suit :

   ```
   cd $COLLATION_HOME/sdk/bin
   ```

   Remarque : Utilisateurs de Windows : les instructions de vérification sur Windows sont identiques, à ceci près que la commande **api.bat** est utilisée à la place de **api.sh** et que le répertoire bin se trouve dans %COLLATION_HOME%\sdk\bin.
2. Affichez l'utilisation de l'interface CLI en lançant la commande suivante :

   ```
   % ./api.sh
   ```
3. Affichez l'état de la reconnaissance en lançant la commande suivante :

   ```
   % ./api.sh -u user -p password -H host discover status
   ```

   Cette commande interroge l'état de reconnaissance actuel. Si vous voyez un état valide (tel que Idle), vous avez réussi à communiquer avec le serveur TADDM et à exécuter une commande.
4. Démarrez une reconnaissance en lançant la commande suivante :

   ```
   % ./api.sh -u user -p password -H host discover start 10.10.10.12
   ```

   Puis vérifiez l'état de reconnaissance pour vérifier que la reconnaissance est en cours :

   ```
   % ./api.sh -u user -p password -H host discover status
   ```
5. Interrogez les portées de reconnaissance en exécutant la commande suivante :

```
% ./api.sh -u user -p password -H host find Scope
```

La commande renvoie les portées définies dans le serveur TADDM au format XML.

6. Collectez tous les systèmes informatiques dans le serveur TADDM à l'aide de la commande suivante :

   ```
   % ./api.sh -u user -p password -H host find ComputerSystem
   ```

   La commande renvoie tous les systèmes informatiques reconnus au format XML.

Utilisation du logiciel SDK TADDM SDK comme composant de logiciel

Pour intégrer le logiciel SDK TADDM dans une application ou dans un environnement de serveur d'applications, définissez les chemins d'accès aux classes d'exécution et de compilation, puis définissez le contrôle d'accès.

Avant de commencer

Les chemins d'accès aux classes pointent vers la bibliothèque Java qui fournit l'interface de programme d'application Java.

La distribution de logiciel SDK TADDM regroupe également les bibliothèques saxon et xalan pour les traitements XSLT et XQuery. Vous pouvez utiliser ces bibliothèques ou vos propres outils de traitement XML pour le traitement XSLT et XQuery.

Procédure

Pour intégrer le logiciel SDK comme un composant, procédez comme suit :

1. Définissez le chemin d'accès aux classes suivant lors de la compilation et de l'exécution :

   ```
   CLASSPATH=$COLLATION_HOME/sdk/lib/taddm-api-client.jar:
     $COLLATION_HOME/sdk/lib/platform-model.jar
   ```

2. Configurez les paramètres d'accès (ID utilisateur et mot de passe).

   Pour utiliser les interfaces de programme d'application Java et CLI, vous devez configurer les paramètres d'accès à l'aide du portail de gestion de données. Vous pouvez utiliser les mêmes ID utilisateur et mot de passe pour accéder à l'interface de programme d'application et au portail de gestion de données.

Que faire ensuite

Après la mise à niveau à partir d'une version précédente de TADDM, vous devrez peut-être mettre à jour le chemin d'accès à la classe pour inclure les fichiers .jar corrects.

Les fichiers .jar du répertoire `$COLLATION_HOME/sdk/lib` sont également utilisés par le serveur TADDM. Par conséquent, le fichier SDK ne doit pas être placé après l'installation. Si vous souhaitez que les fichiers SDK soient dans un emplacement différent, vous pouvez les extraire à partir du fichier `sdk.zip` dans le DVD du produit.

Fichiers .jar Java requis

Les fichiers `taddm-api-client.jar` et `platform-model.jar` sont requis pour l'utilisation de l'interface de programme d'application Java et doivent être présents

dans un répertoire listé dans la variable d'environnement CLASSPATH. Ces fichiers se trouvent dans le sous-répertoire `lib` du répertoire SDK.

Les fichiers `taddm-api-client.jar` et `platform-model.jar` ont remplacé tous les fichiers TADDM JAR précédents à titre d'archives contenant les interfaces de programme d'application client et les définitions de modèle.

Si vous utilisez le kit d'outils XML IBM Tivoli Business Service Manager (TBSM) avec le type de connexion JDBC, vous avez aussi besoin de `oal-topomgr.jar`. Vous pouvez télécharger ce fichier JAR depuis l'emplacement suivant :

`http://`*`taddm.server.machine.name`*`:`*`taddm.server.web.port`*`/GetTaddmVersion/getVersion/getoaltopomgrfile`

Pour détecter les modifications apportées à la version des fichiers JAR sur le serveur TADDM, une application client peut utiliser les adresses URL suivantes afin d'obtenir les valeurs du total de contrôle des fichiers :

- `taddm-api-client.jar`:

  `http://`*`taddm.server.machine.name`*`:`*`taddm.server.web.port`*`/GetTaddmVersion/getVersion/clientjar`
- `platform-model.jar`:

  `http://`*`taddm.server.machine.name`*`:`*`taddm.server.web.port`*`/GetTaddmVersion/getVersion/modeljar`
- `oal-topomgr.jar`:

  `http://`*`taddm.server.machine.name`*`:`*`taddm.server.web.port`*`/GetTaddmVersion/getVersion/oaltopomgrjar`

où *taddm.server.machine.name* correspond au nom de domaine qualifié complet du serveur sur lequel s'exécute TADDM et *taddm.server.web.port* correspond au port HTTP défini pour le serveur TADDM dont la valeur par défaut est 9430.

Remarque : Si le serveur TADDM est démarré dans le cadre de la procédure d'installation, le total de contrôle du fichier `taddm-api-client.jar` est spécifié de manière incorrecte (11111111) par la suite. Si cela se produit, redémarrez le serveur ; les demandes client suivantes renvoient le total de contrôle approprié.

Un client peut également vérifier la version du serveur TADDM à l'aide de l'adresse URL suivante :

`http://`*`taddm.server.machine.name`*`:`*`taddm.server.web.port`*`/GetTaddmVersion/getVersion/taddmversion`

Cette adresse URL renvoie la version du produit (par exemple, `7.2.1`).

Installation et configuration de l'interface de programme d'application SOAP

L'interface de programme d'application SOAP est installée avec le logiciel SDK TADDM. Toutefois, vous devez terminer la procédure décrite dans ce chapitre avant d'utiliser l'interface de programme d'application.

Procédure

Pour terminer l'installation et la configuration de l'interface de programme d'application SOAP, procédez comme suit :

1. Téléchargez le module Axis sur Internet.
2. Décompressez le package dans le répertoire `$COLLATION_HOME/sdk/lib`.
3. Insérez les fichiers JAR du module Axis dans CLASSPATH.

Installation et configuration de l'interface de programme d'application REST

L'interface de programme d'application REST TADDM n'exige pas la spécification de fichiers .jar avec TADDM ; toutefois, il est possible que certains fichiers .jar soient requis si vous voulez travailler avec des objets de modèle TADDM.

Pourquoi et quand exécuter cette tâche

Les fichiers .jar de TADDM vous permettent d'accéder aux classes d'objet de modèle TADDM, ainsi qu'à la classe ModelObjectFactory, qui convertit des objets à partir de/vers des représentations XML.

Procédure

Pour accéder aux classes, procédez comme suit :

Incluez les fichiers .jar appropriés à Java SDK dans un répertoire de la variable d'environnement CLASSPATH :

- `$COLLATION_HOME/sdk/lib/taddm-api-client.jar`
- `$COLLATION_HOME/sdk/lib/platform-model.jar`

Connaissance du Common Data Model

Le Common Data Model (CDM) désigne le langage définitionnel utilisé pour intégrer les connaissances et l'échange de données entre les produits de gestion Tivoli concernant les ressources et les composants d'une entreprise. Le CDM est le modèle utilisé pour communiquer les détails relatifs aux instances de ressource avec la base de données IBM Tivoli Application Dependency Discovery Manager (TADDM).

Il est exclusivement composé de définitions de données. Ces définitions sont des caractéristiques qui identifient les ressources, leur signification et leurs éventuelles restrictions en termes de longueur ou de valeur. Le contenu du CDM est obtenu en fusionnant les informations applicables du secteur d'activité, les normes de modèle de données et les modèles de données utilisés par nos produits actuels en un seul modèle convergent. Il intègre les normes suivantes :

- Modèle CIM Distributed Management Task Force (DMTF)
- Normes de processus métier suivantes :
  - Langage BPEL
  - Spécification IT Infrastructure Library (ITIL)
  - Schéma d'annuaire LDAP
- Normes spécifiques aux domaines suivantes :
  - TeleManagement Forum (TMf),
  - Storage Networking Industry Association (SNIA), etc.

Le Common Data Model est utilisé par plusieurs applications, notamment TADDM. Ces applications peuvent partager des définitions et une terminologie pour les données d'instance de ressource qui leur sont communes, ce qui permet la construction d'applications évoluées englobant l'environnement de gestion tout entier et le partage d'informations entre ces systèmes. Le CDM décrit le contenu d'entrée et de sortie de l'interface de programme d'application TADDM, des détecteurs, des applications d'utilitaire et de reconnaissance de la console de gestion.

Le CDM est différent d'un schéma. En règle générale, un schéma est associé à une base de données, comprend à la fois l'organisation des données en modèle logique et la spécification du mode de stockage de ces données dans des colonnes spécifiques de tables particulières (également appelées modèle physique de la base de données). Le CDM représente un modèle logique composé de définitions qui permettent une identification cohérente des instances de ressource, des informations relatives à ces instances et des relations qui les unissent. Le modèle de données relie les processus métier et les processus informatiques aux systèmes qui les fournissent, aux utilisateurs qui les appellent, aux règles qui les contrôlent, aux ressources utilisées par les processus, etc. Le CDM classe et organise les principales caractéristiques des utilisateurs, les ressources et les informations informatiques métier et il les traite et les présente de manière à ce que toutes les applications puissent les utiliser.

Pour plus de détails sur CDM, consultez les informations suivantes:

- Site Web du modèle de données communes Tivoli, dont l'adresse se trouve dans le répertoire `$COLLATION_HOME/sdk/doc/model`.
- *IBM Tivoli Common Data Model : Guide pour les pratiques recommandées* à l'adresse http://www.redbooks.ibm.com/abstracts/redp4389.html.

Le Common Data Model offre les caractéristiques suivantes :

- Il ne définit pas le schéma physique ni le fonctionnement d'un système de gestion.
- Il définit les ressources et les caractéristiques d'un environnement de gestion surveillées, analysées et contrôlées par un système de gestion.
- Il est également utilisé lorsque des applications de gestion échangent des informations sur les instances de ressource et leurs relations avec d'autres ressources.
- Il normalise les caractéristiques, les types de données utilisés, ainsi que les concepts de classes, d'attributs, d'interfaces, de règles et de stratégies de dénomination.
- Il fournit des définitions cohérentes d'éléments, des valeurs recommandées pour le contenu et des instructions pour le mappage de données d'instance de ressource sur le CDM.

Le Common Data Model se compose des objets suivants :

Classes

Présentent les caractéristiques ou règles suivantes :

- Une classe est une construction utilisée pour regrouper des attributs liés.
- Il s'agit d'une représentation d'un type d'instance de ressource (OperatingSystem, par exemple).
- Considérées comme la structure de base du modèle, les classes contiennent des attributs, installent les interfaces et peuvent également être impliquées dans les relations si vous le souhaitez.
- Les classes sont hiérarchiques et héritent des propriétés des classes parent.
- Les classes peuvent également inclure de façon explicite des propriétés appartenant à un niveau de détail.
- Les instances de classes correspondent aux instances de ressource réelles, les *noms* représentant les ressources physiques ou logiques de l'environnement.

- Les instances possèdent des attributs et peuvent prendre part à des relations. Par exemple, dans un environnement de gestion de base de données, des éléments tels que le serveur de base de données, les tables et les connexions sont des instances.

  Remarque : Les instances peuvent également être des éléments qui ne sont pas seulement gérés, mais participent au processus de gestion, comme les utilisateurs ou les systèmes métier.
- Parmi les différents objets du Common Data Model, les classes sont les seuls utilisés pour représenter les instances de ressource. Certaines classes, mentionnées tout au long de la documentation TADDM, ont une signification particulière :
  - **ModelObject** : cette classe représente la classe de base ou racine du CDM. Toutes les classes sont d'une façon ou d'une autre dérivées de ModelObject. Le terme ModelObject est utilisé dans la documentation pour représenter n'importe quelle classe définie du CDM.
  - **ManagedElement** : antre représentation d'une classe de base ou racine dans le CDM, correspondant directement à la représentation du modèle CIM DMTF de même nom. Le terme ManagedElement est également utilisé dans la documentation pour représenter n'importe quelle classe définie du CDM. Les classes ModelObject et ManagedElement sont utilisées de façon interchangeable.
  - **ManagementSoftwareSystem** - également appelée MSS, cette classe représente les produits de gestion qui fournissent des données à TADDM par le biais d'un mécanisme. Chaque fournisseur de données (y compris les détecteurs de TADDM) est représenté sous forme d'instance de ressource de type ManagementSoftwareSystem.
- Le CDM prend en charge la spécialisation via un héritage simple, bien que l'utilisation d'interfaces confère au modèle certains aspects d'héritage multiple. Toutes les classes sont organisées dans une hiérarchie d'héritage simple monoracine avec la classe **ModelObject** comme racine. Chaque classe, à l'exception de ModelObject, spécifie un seul parent, et la classe enfant hérite de l'ensemble des caractéristiques de la classe parent.
- Le Common Data Model inclut également des règles de nommage pour les objets de modèle qui indiquent les attributs requis pour donner un nom unique aux objets de TADDM. Pour plus d'informations sur les règles de nommage des objets de modèle, voir le chapitre **Désignation des instances**.
- Classes permanentes et non permanentes :
  - Une classe permanente est une classe dont les instances peuvent être enregistrées dans une base de données, alors que celles d'une classe non permanente ne peut pas être enregistrées dans une base de données.
  - Si vous utilisez le langage MQL (Model Query Language), vous ne pouvez lancer de requête que sur des classes permanentes. La seule exception concerne une requête sur l'attribut "guid" d'une valeur ModelObject (classe non permanente), comme dans l'exemple suivant :
    - L'attribut, "source", est un ModelObject et les requêtes suivantes renvoient le même résultat :

      ```
      SELECT * FROM TransactionalDependency WHERE source.guid ==
      'E72B13789C9039BFB32E3822FE50C197'
      ```

```
SELECT * FROM TransactionalDependency WHERE source ==
'E72B13789C9039BFB32E3822FE50C197'
```

– Dans la Javadoc du modèle (Javadoc pour CommonDataModel de TADDM), si la balise, '**Persistable**', est définie sur `true` pour une classe donnée, la classe est alors persistante. Si la balise n'est pas présente pour une classe donnée, il s'agit d'une classe non permanente.
  - Exemples de classes permanentes : ComputerSystem, SoftwareModule, AppServer
  - Exemples de classes non permanentes : ModelObject, Database, LogicalElement

Attributs

Présentent les caractéristiques ou règles suivantes :

- Un attribut définit une propriété particulière valide pour une classe.
- Chaque attribut possède une signification ou une sémantique particulière en termes de contenu prévu.
- Les attributs sont indiqués sur les classes du CDM ainsi que sur les interfaces.
- Les instances des attributs sont les *adjectifs* qui décrivent les caractéristiques des instances et servent à différencier les instances d'une même classe, telles que les différents **fabricants** des instances de la classe **ComputerSystem**.
- Lors de la création d'une instance de ressource, il est possible de stocker les données d'un attribut valide pour une instance de ressource.
- Tous les attributs ne doivent pas nécessairement contenir de valeur ; en revanche, certains attributs sont utilisés pour représenter une identité unique d'une instance de ressource. Ces attributs sont souvent appelés *attributs d'identité*.

Interfaces

Facilitent la réutilisation d'un ensemble d'attributs et améliorent la souplesse de définition des relations. Par exemple, l'attribut **VersionString** est valide pour plusieurs types d'instance de ressource (classes) différents. Plutôt que de dupliquer cet attribut dans plusieurs classes du CDM, une interface est créée pour représenter l'ensemble des attributs appartenant aux données de version.

Les instances de ressource ne peuvent pas reposer sur une interface. Toute classe qui implémente automatiquement une interface reçoit l'ensemble d'attributs et de relations de l'interface comme s'il existait dans la classe. Les interfaces sont hiérarchiques et peuvent dériver leurs attributs et relations de l'interface parent par héritage.

Une des interfaces mentionnées dans la documentation TADDM possède une signification particulière. Elle est appelée *élément de configuration*. Cette interface est utilisée pour désigner des classes particulières du CDM, dont les instances servent d'élément de configuration défini par le terme ITIL correspondant. Certaines classes du CDM, comme les données financières, ne sont pas définies pour servir d'éléments de configuration, car le CDM représente des aspects de différents environnements.

Relations

Présentent les caractéristiques ou règles suivantes :

- Associations entre deux instances de ressource, qui montrent comment ces instances sont liées l'une à l'autre.

- Les relations peuvent uniquement être établies entre des classes, qu'elles soient de même type ou de types différents.
- Chaque relation possède une définition ou un type particulier. Ces différents types de relation ont une sémantique particulière, propre au type d'association entre les instances de ressource.

  Par exemple, **gère** est l'un des types de relation du CDM. Il représente l'instance source participant à un rôle de contrôle de l'instance de ressource cible de la relation. **installedOn** est un autre type de relation, qui représente l'instance source sous forme d'objet installé sur l'instance de ressource cible. Ces deux relations peuvent être appliquées à des instances de ressources où la source est une instance de la classe **Agent** et la cible, une instance de la classe **OperatingSystem**. Elles ont toutefois une signification très différente. Plusieurs relations peuvent être établies entre deux classes (et deux instances de ressource). Chaque relation constitue une association entre deux instances.

  Dans le CDM, chaque instance de relation possède une source et une cible, qui sont les rôles de la relation. Le nombre d'instances pouvant prendre part à chaque rôle est important. Certaines relations n'autorisent la participation que d'une seule instance, tandis que d'autres autorisent un nombre d'instances illimité. Le nombre d'instances pouvant participer à chaque rôle est appelé *cardinalité de la relation*.

Types de données

Les informations contenues dans les attributs et les mesures doivent être présentées dans une syntaxe connue. C'est pourquoi le CDM définit un ensemble de types de données à utiliser pour représenter les informations relatives aux entités.

Les types de données définis dans le modèle ne constituent pas une représentation physique des données, mais en indiquent la longueur et parfois, le codage ou la valeur recommandée pour le contenu des données.

Le modèle inclut également des types de données énumérés qui permettent aux produits de comprendre la signification commune de certaines valeurs.

Désignation des instances

Les noms (ou attributs de nommage) constituent la base de l'identification des ressources et de la synchronisation entre les instances de ressource représentant un même objet dans le centre de données.

La dénomination repose sur la génération, l'utilisation et le partage d'attributs lisibles par l'utilisateur à des fins d'identification des instances de ressource. Un nom unique est créé pour une instance de ressource en regroupant le contenu d'attributs particuliers pour cette instance. Selon la taille du modèle de données, il existe de nombreuses façons de nommer une instance de ressource. Pour organiser la méthode de génération d'un nom unique, le Common Data Model utilise le concept de règles de nommage afin de regrouper les différents attributs qui constituent une identité unique.

Règles de nommage

Une règle de dénomination indique comment nommer les instances d'une classe particulière, telles que des ressources, des personnes et des systèmes.

Les règles de nommage contiennent un ensemble d'attributs obligatoires pour nommer une ressource donnée. En général, une règle de dénomination a pour but de regrouper des attributs afin de constituer une identité unique. Si deux instances

portent le même nom, elles sont considérées comme faisant référence à la même entité. Par exemple, différentes entités d'une couche réseau niveau 2 sont souvent identifiées de la même manière, à l'aide d'une adresse MAC, même si les entités sont des instances de classes différentes, peut-être non associées. Les adresses MAC, par leur structure, présentent un espace par lequel tous les noms valides d'une station d'une couche réseau niveau 2 peuvent être affectés.

Remarque : Cela ne dépend pas du type de réseau impliqué, qui pourrait être 10-BaseT, 1000-BaseT ou anneau à jeton.

Il existe deux cas particuliers dans lesquels les règles de nommage ne contiennent pas simplement des attributs.

1. Contexte d'affectation de nom :

   Lors de la dénomination d'une instance de ressource, il arrive que peu d'informations soient disponibles pour nommer l'instance de façon unique à partir des attributs existant sur la classe. Dans de tels cas, certaines règles de nommage indiquent une relation en plus de l'ensemble d'attributs obligatoire. Ces relations appliquent ce que l'on appelle un *contexte d'affectation de nom* sur l'instance de ressource et exigent l'utilisation d'une seconde ressource pour identifier une autre instance de ressource de façon contextuelle.

   Par exemple :

   - La seule information connue sur une instance particulière d'un système d'exploitation est le type du système d'exploitation.
   - L'attribut représentant le type d'un système d'exploitation n'est pas suffisamment unique pour créer une instance de ressource unique représentant le système d'exploitation.
   - Pour utiliser cet attribut, la règle de dénomination indique une relation **installedOn** obligatoire entre l'instance du système d'exploitation et une instance d'un système informatique (il est implicitement nécessaire de créer également une instance valide d'un système informatique pour pouvoir établir la relation).

2. NON :

   Certaines règles de nommage sont appliquées avec un ensemble d'attributs défini acceptable pour nommer une instance de ressource de façon unique dans la majorité des cas. Toutefois, dans certains cas, une autre règle de dénomination est nécessaire dans le Common Data Model pour détailler davantage l'identité d'une ressource, en utilisant l'ensemble d'attributs employé par une autre règle de dénomination, mais en y ajoutant des attributs supplémentaires.

   Dans la mesure où la méthode de création d'une instance unique repose sur le respect des règles de nommage, une règle de dénomination présentant des exigences moins spécifiques pour générer une identité n'est pas souhaitable lorsque des attributs plus spécifiques sont fournis. Pour empêcher que la règle de dénomination moins spécifique soit employée, certaines règles de nommage utilisent une instruction OmittedIdentifier sur un attribut. Elle est également appelée «NON» dans la section du site Web du Common Data Model portant sur les règles de nommage.

   Remarque : L'adresse de ce site Web est indiquée dans le répertoire `$COLLATION_HOME/sdk/doc/model`.
   Lorsque cette opération NON est mentionnée, elle indique que la valeur de l'attribut doit être NULL. Si l'attribut mentionné dans l'opération OmmittedIdentifier (NON) comporte un contenu, la règle de dénomination n'est pas utilisée pour identifier une ressource de façon unique. Par exemple :

- Il existe une règle de dénomination nommée ActivityName sur la classe Activity.
- Cette règle de dénomination exige que l'attribut ActivityName contienne une valeur.
  - Avec cette règle de dénomination particulière, les noms de la classe Activity sont supposés être globalement uniques dans l'environnement du client.
- Dans les cas où les noms de classe Activity ne sont pas uniques, il existe une seconde règle de dénomination appelée QualifiedActivity.
  - Cette règle requiert l'attribut ActivityName et une relation est propriétaire de entre une instance de la classe OrganizationalEntity et l'instance de la classe Activity.
- Dans la mesure où les règles de nommage utilisent un attribut commun, ActivityName, et où une règle de dénomination constitue un degré de précision supplémentaire par rapport à une autre règle de dénomination, une seule règle doit être utilisée pour nommer l'instance d'Activity.
- Par conséquent, la règle de dénomination ActivityName a appliqué l'opération NON à la relation est propriétaire de. Cela signifie que la relation est propriétaire de ne doit pas être remplie pour que la règle de dénomination ActivityName puisse être utilisée.

L'identification repose sur la génération, l'utilisation et le partage d'une valeur concise et unique assimilable par une machine à des fins d'identification des traitements. Les instances de ressource représentées par le Common Data Model portent à la fois des noms et des identificateurs :

- Les noms sont des chaînes plus longues, essentiellement alphabétiques, dont l'utilisateur se sert pour faire référence aux entités.
- Les identificateurs sont des valeurs plus courtes, denses et essentiellement numériques que le système de gestion utilise pour identifier les entités de manière unique.

Identificateurs globaux uniques TADDM

Les valeurs d'identification sont généralement appelées identificateurs globaux uniques (GUID - globally unique identifier). Les identificateurs globaux uniques TADDM sont créés selon la spécification des identificateurs uniques universels UUID version 3 (IETF Standards Track RFC 4122) et sont utilisés comme des identificateurs d'éléments de configuration (EC).

Les identificateurs de version 3 sont générés par le traitement d'une chaîne par un algorithme de cryptographie de type MD5. TADDM transmet au composant de génération de l'identificateur global unique une chaîne constituée des valeurs des attributs utilisés dans les règles de nommage. La plupart des éléments de configuration disposent de plusieurs règles de nommage et peuvent donc générer plusieurs identificateurs globaux uniques. Les valeurs d'attributs disponibles à la création de l'EC déterminent les identificateurs globaux uniques générés. En général, le premier identificateur global unique qui est généré pour un objet est considéré comme l'identificateur global unique maître ou l'identificateur principal de cet objet. Les autres identificateurs globaux uniques sont des alias de cet identificateur principal.

Si les EC sont reconnus avec les mêmes attributs et valeurs, ils possèdent toujours le même ensemble d'identificateurs globaux uniques. Cependant, le premier identificateur global unique qui devient ultérieurement un identificateur global unique maître est généré de manière aléatoire. C'est la raison pour laquelle un EC

précis peut ne pas avoir le même identificateur global unique maître dans différentes installations TADDM. De la même manière, il peut de pas être à nouveau sélectionné lorsque l'élément est supprimé ou que la base de données est recréée. Les mêmes types d'EC, tels que ComputerSystems, peuvent également utiliser les identificateurs globaux uniques calculés à partir d'une autre règle de nommage que leurs identificateurs globaux uniques maîtres.

En général, les interfaces de programme d'application TADDM (API) identifient les EC par leurs identificateurs globaux uniques maîtres, mais elles peuvent également les identifier par leurs alias. C'est pourquoi, si vous souhaitez rechercher un EC particulier, vous pouvez le rechercher en utilisant son identificateur global unique d'alias.

Endommagement de l'identificateur global unique

Les identificateurs globaux uniques qui sont des alias d'un identificateur global unique maître risquent de s'endommager au cours du cycle de vie d'un élément de configuration. L'endommagement se produit lorsqu'un attribut définissant une règle de nommage unique, comme une signature, change. Après ce changement, un nouvel ensemble d'identificateurs globaux uniques est généré et remplace les anciennes valeurs. Si les attributs d'un identificateur global unique maître changent, cet identificateur global unique reste identique et un nouvel alias est ajouté.

Changements de l'identificateur global unique maître

Un identificateur global unique maître d'un élément de configuration particulier peut changer en fonction de l'une des conditions suivantes :

Suppression d'un élément de configuration et nouvelle reconnaissance
Lorsqu'un élément de configuration est supprimé de la base de données TADDM, un autre identificateur global unique peut être sélectionné comme identificateur global unique maître au cours du stockage suivant de cet EC.

Scénario de fusion des éléments de configuration
Lorsque de nouvelles données sont disponibles dans TADDM, deux EC différents peuvent être identifiés comme étant de la même instance. Un utilisateur peut également lancer la fusion manuellement. Dans ce scénario, les attributs d'un EC transitoire et d'un EC définitif peuvent fusionner. Par conséquent, l'identificateur global unique maître d'un EC transitoire devient un nouvel alias de l'élément définitif et l'identificateur global unique maître de l'EC définitif représente un EC créé après la fusion.

Mise à niveau de TADDM
Lors de la mise à niveau sur une nouvelle version de TADDM, les attributs compris dans les règles de nommage peuvent changer. Cette situation peut également affecter le processus de migration de données supposé vérifier que les identificateurs globaux uniques maîtres restent identiques après la mise à niveau. Une nouvelle version de détecteurs ou de détecteurs Discovery Library Adapters peut également changer le mode de stockage des valeurs d'attributs.

Noms de classe

Les noms d'objet de classe du Common Data Model TADDM peuvent être référencés sous leur forme longue ou sous leur forme abrégée. La plupart des noms d'objet peuvent être référencés par leur forme courte.

Pour un système informatique, le nom abrégé est ComputerSystem et le nom long, com.collation.platform.model.topology.sys.ComputerSystem.

Les doubles constituent une exception à l'utilisation des noms abrégés. Par exemple, SSLSettings doit être référencé par son nom long car il existe deux instances de SSLSettings:

- `com.collation.platform.model.topology.app.lotus.SSLSettings`
- `com.collation.platform.model.topology.app.SSLSettings`.

L'exemple de code suivant contient tous les noms abrégés et longs des classes du Common Data Model (CDM). Une fois cette commande exécutée, la liste des noms de classe en double qui doivent être référencés sous leur nom long s'affiche à la fin des résultats.

```
Exemple DisplayClassNames
import com.collation.proxy.api.client.*;
import com.collation.proxy.api.util.*;
import com.ibm.cdb.api.ApiFactory;
import java.util.*;

class DisplayClassNames {

    public static void main(String[] args) {
        CMDBApi api = null;

        try {
            System.out.println("--- Affichage des noms d'objet modèle ----" );

            ApiConnection conn = ApiFactory.getInstance().
                 getApiConnection("localhost", -1, null, false);
            ApiSession sess1 = ApiFactory.getInstance().getSession(conn,
              "administrator",
 "collation", ApiSession.DEFAULT_VERSION);
            api = sess1.createCMDBApi();

            String[] classNameArray = api.getClassNames();

            ArrayList shortNames = new ArrayList(classNameArray.length);
            ArrayList dups = new ArrayList(10);
            for (int i = 0; i < classNameArray.length; i++) {
               // imprimer les noms de classe abrégés et longs
               System.out.println ("\nNom abrégé = " + classNameArray[i]);
               System.out.println ("Nom long = " + classNameArray[i+1]);
               // Vérifier si le nom abrégé est un double
               if (shortNames.contains(classNameArray[i])) {
                  dups.add(classNameArray[i]);
               } else {
                  shortNames.add(classNameArray[i]);
               }
               i++;
            }
            System.out.println("\nLes classes suivantes doivent être spécifiées sous
              leur nom long : ");
            System.out.println(dups);
            sess1.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            if (api != null) {
              try {
                 api.close();
              } catch (Exception e) { }
            }
        }
```

```
    }
}
```

Dépendances entre les ressources

TADDM reconnaît et classe les différents types de dépendances sur plusieurs niveaux, ces dépendances sont reflétées dans le CDM. Les dépendances modèlent les relations d'exécution entre les différents composants du CDM.

Il existe plusieurs types de dépendances, notamment :

- Les dépendances transactionnelles

  Les dépendances transactionnelles surviennent entre des composants d'application, tels que les serveurs Web, les serveurs d'application et les bases de données. Le composant dépendant émet des requêtes au composant fournisseur pour effectuer certaines fonctions. Par exemple, une connexion Java Database Connectivity (JDBC) depuis un serveur Java 2 Platform, Enterprise Edition (Java EE) vers une base de données représente une dépendance transactionnelle. Dans ce cas, le fournisseur est souvent appelé serveur et le composant dépendant consommateur ou un client.
- Les dépendances de service

  Les dépendances de service surviennent entre des composants d'application et des services d'infrastructure, tels que DNS (Domain Name System), LDAP (Lightweight Directory Access Protocol) et NFS (Network File System). Le fournisseur est le service d'infrastructure et le composant dépendant requiert des services système du fournisseur. Par exemple, une demande de mappage d'un nom DNS vers une adresse IP.
- Les dépendances IP

  Les dépendances IP se produisent entre deux systèmes informatiques ou entre un serveur d'application et un système informatique. TADDM crée ce type de relation lorsqu'il reconnaît une relation entre deux systèmes informatiques sans parvenir à découvrir exactement quel serveur d'application est impliqué.
- Les dépendances du système

  Les dépendances du système se produisent entre un serveur d'application et son système informatique hôte.
- Les dépendances Application à application

  Les dépendances Application à application ont lieu entre deux applications métier.

Exemple de dépendances

Lorsque des dépendances de transaction sont créées pour deux serveurs d'application, aucune dépendance IP n'est créée entre eux. De même, aucune dépendance IP n'est créée entre le serveur d'application et ses hôtes. Toutefois, il peut exister une autre connexion logique par exemple entre deux processus et en fonction de ces connexions, des dépendances IP peuvent être créées entre des systèmes informatiques. Par exemple, prenez en considération le scénario suivant :

- Le système informatique (CS1) héberge un serveur d'application (AP1) et un processus (P1)
- Le système informatique (CS2) héberge un serveur d'application (AP2) et un processus (P2)

  Il existe deux connexions logiques créées par TADDM : AP1 <-> AP2 et P1<->P2

Dans ce scénario, une dépendance transactionnelle est créée entre AP1 et AP2 (en fonction de la connexion logique AP1 <-> AP2). Une dépendance IP est créée entre CS1 et CS2 (en fonction de la connexion logique entre P1<->P2).

Simplified Model

Au vu des problèmes causés par le Common Data Model, un nouveau Simplified Data Model pour le stockage de données a été introduit dans la version 7.3 de TADDM. Les classes sont les seuls éléments conservés de l'ancien modèle.

Le nouveau Simplified Model facilite la création de détecteurs personnalisés basés sur un script et permet d'étendre la portée de reconnaissance. Vous pouvez personnaliser de nouveaux détecteurs et éditer les détecteurs existants.

Remarque : Le Simplified Model n'est pas pris en charge par les autres produits que vous intégrez à TADDM.

Le tableau suivant répertorie les principales fonctions introduites par le Simplified Model et indique les rubriques où vous pouvez obtenir des informations supplémentaires.

Tableau 5. Fonctions du Simplified Model

Fonction	Emplacement
des classes génériques de niveau supérieur dans la hiérarchie qui représentent les éléments de configuration principaux ;	«Package, classes et hiérarchies»
des nouveaux objets de niveau moyen qui représentent les composants déployables ;	«Package, classes et hiérarchies»
un mécanisme permettant d'étendre les nouveaux niveaux supérieurs avec un nombre illimité d'attributs (attributs étendus) ;	«Attributs étendus», à la page 23
un mécanisme permettant de stocker des blocs entiers de données brutes (par exemple, des données XML ou des résultats de commande) et de connecter ceux-ci aux éléments de configuration de niveau supérieur (instances étendues).	«Instances étendues», à la page 31
nouvel opérateur `eval` utilisé dans les requêtes MQL pour les attributs étendus et les instances étendues.	«Présentation du langage MQL», à la page 43
nouvel attribut de règle de nommage générique `openId`.	«Attribut de règle de nommage générique `OpenId`», à la page 21

Package, classes et hiérarchies

Package et classes

Toutes les nouvelles classes sont stockées dans le package `com.collation.platform.model.topology.simple`. Les noms des classes commencent par la lettre majuscule "S" afin d'éviter des conflits car TADDM distingue les types de données par leurs noms abrégés.

Attributs de hiérarchie

Chaque objet de modèle de données contient deux attributs de hiérarchie, `hierarchyDomain` et `hierarchyType`, qui définissent les informations qui, dans l'ancien modèle, étaient présentes dans le package et le type de

chaque classe. Par exemple, le type ComputerSystem contenait beaucoup de systèmes informatiques caractéristiques qui représentaient différents systèmes d'exploitation tels que sys.linux.LinuxUnitaryComputerSystem ou sys.windows.WindowsComputerSystem. Ces objets doivent être stockés séparément. Le nouveau Simplified Model permet de stocker des objets des deux types comme le type SComputerSystem, lorsque ces deux attributs sont définis comme suit :

- Pour sys.linux.LinuxUnitaryComputerSystem :

```
hierarchyDomain="sys.unix.linux"
hierarchyType="RedHat"
```

- Pour sys.windows.WindowsComputerSystem :

```
hierarchyDomain="sys.windows"
hierarchyType="Windows7"
```

Les valeurs de l'attribut hierarchyDomain indiquent les niveaux de domaines, du niveau le plus général au niveau le plus spécifique. Par exemple, dans la valeur "app.db.mongodb", app correspond à un serveur d'applications, db correspond au serveur de base de données et mongodb correspond à une base de données spécifique (dans ce cas, MongoDB).

Les attributs de hiérarchie permettent de gérer intégralement l'interface utilisateur. Ils sont également utilisés dans la formulation de requêtes.

Nouveaux types de hiérarchie

La liste suivante présente les nouveaux types de hiérarchie :

- SComputerSystem - remplace la hiérarchie ComputerSystem ;
- SSoftwareServer - remplace la hiérarchie AppServer ;
- SLogicalGroup - remplace la hiérarchie AppServerCluster et représente les collections qui sont stockées par un détecteur, quelle que soit leur nature (par exemple, des clusters ou des domaines) ;
- SFunction - remplace la hiérarchie Function et représente des règles et des fonctions supplémentaires.
- rSSoftwareInstallation - représente les progiciels physiques présents sur un système informatique qui constitue un serveur logiciel (bibliothèques créées par le fournisseur) ;
- SPysicalFile - remplace les hiérarchies AppConfig et LogicalContent et représente un fichier de configuration. L'intégralité du contenu de ce fichier est capturée par une reconnaissance.
- SDeployableComponent - un nouveau type qui représente les types considérés comme des sources ou cibles de relation, qui sont déployées sur les systèmes informatiques, les serveurs logiciels ou les groupes logiques. Il s'agit d'objets de niveau moyen. Certains des anciens types de modèles CDM sont encore rattachés à la hiérarchie SDeployableComponent afin de garantir la compatibilité avec les versions antérieures. La liste suivante indique les types en question :
  - BiztalkApplication
  - Database
  - DominoDatabase
  - ExchangeLink, ExchangeStorageGroup
  - Système de fichiers
  - IIsWebServer, IIsWebVirtualDir
  - MBExecutionGroup, MBMessageFlow, MessageBox
  - MQChannel, MQQueue

- OracleSchema
- SharePointWebApplication
- SoftwareModule - le seul composant du type SoftwareModule qui est conservé est J2EEApplication car il s'agit du seul composant déployable dans ce type.
- WebVirtualHost

Tous ces nouveaux types héritent des types abstraits suivants qui sont introduits dans le modèle de données à des fins de modélisation : SStandaloneObject, SContextualObject, SGroup.

Attribut lastStoreTime
L'attribut lastModifiedTime est déprécié car son nom prête à confusion. Un nouvel attribut lastStoreTime le remplace.

Migration

La migration depuis l'ancien modèle vers le nouveau modèle est automatique. Aucune tâche supplémentaire n'est requise. Les requêtes MQL et les vues SQL sont compatibles avec la version antérieure.

Modifications du portail de gestion de données

La sous-fenêtre Récapitulatif de l'inventaire de l'onglet **Inventaire** contient un nouveau type de composant appelé Données génériques. Celui-ci contient des objets de classes stockées dans le package simple. Vous pouvez également parcourir des dossiers génériques dans la sous-fenêtre Composants reconnus et afficher les détails des objets reconnus pour les nouveaux types de classe. Pour plus d'informations sur l'affichage des attributs étendus et des instances étendues, voir «Attributs étendus», à la page 23 et «Instances étendues», à la page 31.

Applications métier

Le Simplified Model est pris en charge par le moteur d'applications métier. Vous pouvez inclure les nouvelles classes, les nouveaux attributs et l'opérateur eval dans les requêtes lorsque vous utilisez les applications métier. Vous pouvez également créer des applications métier à partir des objets stockés à l'aide du nouveau modèle.

Attribut de règle de nommage générique OpenId

Pour chaque projet, vous devez spécifier des attributs et définir une règle de nommage indiquant les attributs qui fournissent une valeur unique. Les règles de nommage varient selon le type de hiérarchie. Les nouveaux types génériques qui sont stockés dans le package simple, sont stockés avec les mêmes règles de nommage que dans l'ancien CDM. Les objets qui sont issus du type SStandaloneObject sont stockés avec l'attribut openId. Les objets qui sont issus du type SContextualObject sont stockés avec les attributs context et scopedId.

Les attributs openId et scopedId sont tous deux du type chaîne dans une base de données mais sont d'un nouveau type OpenId Java au niveau de l'interface de programme d'application. Par conséquent, ces attributs ont un format spécifique, aligné sur le schéma OpenId. Le type OpenId contient des méthodes permettant de construire facilement une chaîne significative représentant une valeur d'une règle de nommage. Les identificateurs globaux uniques sont calculés en fonction des valeurs indiquées dans l'attribut openId.

Exemples

Exemple 1

Pour les serveurs, la règle de nommage est généralement basée sur deux attributs. Il s'agit du point principal d'accès au service créé à partir de l'adresse IP principale du serveur et d'un port sur lequel ce service écoute. L'attribut `openId` est indiqué de la manière suivante :

```
id = OpenId().addId('IP' , seed.getPrimaryIpAddress().getStringNotation())
.addId('port' , str(seed.getPort()))
```

Exemple 2

Dans le Common Data Model, le type `ComputerSystem` ancien dispose d'une règle de nommage fondée sur les attributs `manufacturer`, `model`, et `serialNumber`. Ces trois attributs sont définis dans la classe de manière explicite et lorsque leurs valeurs sont définies, un objet `ComputerSystem` peut être stocké.

```
LinuxUnitaryComputerSystem cs = ModelFactory.newInstance
(LinuxUnitaryComputerSystem.class);
cs.setManufacturer("RedHat");
cs.setModel("Linux");
cs.setSerialNumber("as00123012");
```

Un objet disposant du mappe d'attributs suivant est stocké dans la couche de persistance :

```
manufacturer -> RedHat
serialNumber -> as00123012
isPlaceholder -> false
model -> Linux
```

Dans le nouveau Simplified Model, le stockage d'objets du type `SComputerSystem` simplifié avec des valeurs identiques est semblable. Cependant, si un autre attribut de règle de nommage est disponible pour le système informatique en question, par exemple l'ID qu'un cluster affecte à chaque système informatique physique qu'il gère, et que cet attribut de règle de nommage n'est pas défini dans le modèle, alors il peut être étendu en utilisant le type `OpenId`. Lorsque l'attribut est étendu, il est stocké mais représente également de manière unique le système informatique. Le type `OpenId` est utilisé de la manière suivante :

```
SComputerSystem scs = ModelFactory.newInstance(SComputerSystem.class);
scs.setHierarchyDomain("sys.unix.linux");
scs.setHierarchyType("RedHat");
scs.setManufacturer("RedHat");
scs.setModel("Linux");
scs.setSerialNumber("as00123012");
OpenId id = new OpenId();
id.addId("clusterInternalId", "66");
scs.setOpenId(id);
```

Exemple 3

Le type `OpenId` peut également être défini comme suit :

```
scs.setOpenId(new OpenId().addId("clusterInternalId", "66"));
```

Le mappe d'attributs suivant est stocké :

```
manufacturer -> RedHat
serialNumber -> as00123012
model -> LinuxhierarchyType -> RedHat
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
openId -> <openId><id><name>clusterinternalid</name><value>66</value></id>
</openId>
```

Exemple 4

L'ajout d'une fonction à ce système informatique simplifié est très semblable. L'exemple suivant présente la manière de créer l'attribut `OpenId` pour des valeurs déjà définies pour les attributs de classe simplifiés :

```
SFunction sf = ModelFactory.newInstance(SFunction.class);
sf.setName("Cisco Firewall");
sf.setHierarchyDomain("function.net.firewall");
sf.setHierarchyType("Cisco");

OpenId fid = new OpenId(sf);
fid.addId("name", null);
fid.addId("type", "firewall");
sf.setScopedId(fid);
sf.setProvider(scs);
```

Le mappe d'attributs suivant est stocké :

```
hierarchyType -> Cisco
isPlaceholder -> false
provider -> {hierarchyType=RedHat;hierarchyDomain=sys.unix.linux;
isPlaceholder=false;openId=<openId><id><name>clusterinternalid</name>
<value>66</value></id></openId>;}
hierarchyDomain -> function.net.firewall
name -> Cisco Firewall
scopedId -> <openId><id><name>name</name><value>Cisco Firewall</value>
</id><id><name>type</name><value>firewall</value></id>
</openId>
```

Exemple 5

Pour créer facilement un ID simple sans aucune distinction quant aux attributs spécifiques qu'il contient, définissez l'attribut comme suit :

```
OpenId fid = new OpenId(sf);
sf.setProvider(scs);
sf.setScopedId(new OpenId().addId("id19921"));
```

Le mappe d'attributs suivant est stocké :

```
hierarchyType -> Cisco
isPlaceholder -> false
provider -> {hierarchyType=RedHat;hierarchyDomain=sys.unix.linux;
isPlaceholder=false;openId=<openId><id><name>clusterinternalid</name>
<value>66</value></id></openId>;}
hierarchyDomain -> function.net.firewall
name -> Cisco Firewall
scopedId -> <openId><id><name>id</name><value>id19921</value></id></openId>
```

Concepts associés:

«Attributs étendus»
Dans l'ancien Common Data Model, les attributs étendus étaient utilisés pour définir manuellement jusqu'à 100 attributs par type de CDM pour enregistrer les données. Ils permettaient d'étendre des types de CDM avec des attributs extérieurs au domaine, comme un numéro de salle de serveur, et de développer la portée de la reconnaissances des éléments de configuration. Dans le nouveau Simplified Model, les attributs étendus permettent d'enregistrer tous les attributs simples des classes CDM qui étaient précédemment présentes dans les types inférieurs en hiérarchie.

Attributs étendus

Dans l'ancien Common Data Model, les attributs étendus étaient utilisés pour définir manuellement jusqu'à 100 attributs par type de CDM pour enregistrer les données. Ils permettaient d'étendre des types de CDM avec des attributs extérieurs au domaine, comme un numéro de salle de serveur, et de développer la portée de

la reconnaissances des éléments de configuration. Dans le nouveau Simplified Model, les attributs étendus permettent d'enregistrer tous les attributs simples des classes CDM qui étaient précédemment présentes dans les types inférieurs en hiérarchie.

Par exemple, dans l'ancien modèle, le type `ExchangeServer` possède l'attribut `productID` du type de chaîne défini. Dans le nouveau modèle, le type `ExchangeServer` est enregistré de la manière suivante :

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));
```

L'attribut `productID` ne peut pas être enregistré parce que le type `SSoftwareServer` n'enregistre que des attributs essentiels et ne peut pas être étendu. Les attributs étendus permettent d'enregistrer de tels attributs spécifiques.

Type de modèle de données

Les changements suivants ont été introduits dans le nouveau modèle de données :

- Les attributs étendus sont enregistrés avec les éléments de configuration dans l'attribut `XA` d'un nouveau type `ExtendedAttributesData` personnalisé. Les données conservées dans des objets séparés du type `UserData` sont migrées dans l'attribut `XA`.
- La limitation d'enregistrement de jusqu'à 100 attributs par type de CDM a été supprimée. Le nombre d'attributs pouvant être enregistrés dans un objet unique dépend de la capacité du type colonne XML de la base de données. En outre, vous pouvez utiliser l'option `-g` du programme de chargement en bloc pour enregistrer des attributs étendus.
- Les attributs étendus ont des catégories. Si aucune catégorie n'est sélectionnée, un attribut est enregistré dans la catégorie par défaut `Général`. Tous les attributs étendus présents dans l'ancien modèle de données ont été déplacés dans la catégorie par défaut `Général`.
- L'ancien attribut `byte[] extendedAttributes` n'est conservé que pour assurer la compatibilité avec des versions antérieures et est déprécié. Les méthodes d'interface de programme d'application (API) publiques `setExtendedAttributes` et `getExtendedAttributes` sont dépréciées également.

Affichage des attributs étendus

Après avoir exécuté la reconnaissance d'un détecteur pour lequel vous avez spécifié l'attribut `XA`, vous pouvez afficher les attributs étendus dans le portail de gestion de données. Ouvrez la sous-fenêtre Détails d'un objet reconnu. L'onglet **Attributs étendus** n'existe plus ; les informations sur les attributs étendus de la catégorie par défaut sont affichées dans l'onglet **Général**. Les attributs étendus d'une catégorie personnalisée sont affichés dans un onglet personnalisé. Par exemple, les attributs étendus de la catégorie Markers sont affichés dans l'onglet **Markers**.

Exemple d'utilisation dans un détecteur

L'exemple suivant présente le type `ExtendedAttributesData` qui est utilisé pour enregistrer un attribut étendu ou un attribut avec un élément de configuration. L'attribut `productID` est conservé dans une catégorie par défaut. Une catégorie relative à un emplacement physique d'un serveur de logiciels est créée.

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
xa.addAttribute("productID", "ID12021");
xa.addAttribute("Location", "buildingNo", "North23");
xa.addAttribute("Location", "floorNo", "3");
xa.addAttribute("Location", "roomNo", "15");
xa.attachTo(sr);
```

Remarques :

- Dans TADDM 7.3.0 et 7.3.0.1, deux méthodes permettent de stocker les attributs étendus. La première, `attachTo` est utilisée dans l'exemple précédent. Cette méthode doit être spécifiée après toutes les entrées `addAttribute()`. L'autre possibilité consiste à utiliser la méthode `setXA`, par exemple :

  ```
  xa.addAttribute("Location", "roomNo", "15");
  xa.toXML();
  sr.setXA(xa);
  ```

  Lorsque vous utilisez la méthode `setXA`, vous devez également spécifier la méthode `toXML` pour convertir les attributs étendus dans un agencement qui pourra être enregistré.

- Fix Pack 2 Dans TADDM 7.3.0.2, les méthodes `attachTo` et `setXA` s'utilisent de la même façon. Aussi, il n'est pas nécessaire d'appeler la méthode `toXML` pour les utiliser.

Le mappe d'attributs suivant est stocké :

```
hierarchyType -> Exchange
isPlaceholder -> false
openId -> <openId><id><name>servername</name><value>Exchange1122</value>
</id></openId>
hierarchyDomain -> app.messaging.exchange
XA -> <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
  <attribute name="productID" category="taddm_global">ID12021</attribute>
  <attribute name="buildingNo" category="Location">North23</attribute>
  <attribute name="floorNo" category="Location">3</attribute>
  <attribute name="roomNo" category="Location">15</attribute>
</xml>
```

Mise à jour partielle

Un élément de configuration peut être enregistré avec les mêmes valeurs des règles de nommage à partir de différentes sources de données, et en conséquence les valeurs sont fusionnées dans un objet. Pour éviter cela, nous utilisons un mécanisme de mise à jour partielle pour fusionner deux attributs `XA` formatés XML. Par exemple, si une source est enregistrée avec objet 1 et une autre source est enregistrée avec objet 2, un élément de configuration qui détient l'attribut fusionné est créé.

Objet 1

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
```

```
xa.addAttribute("productID", "ID12021");
xa.addAttribute("internal", "ID1233");
xa.addAttribute("Location", "buildingNo", "North23");
xa.attachTo(sr);
```

Objet 2

```
SSoftwareServer sr = ModelFactory.newInstance(SSoftwareServer.class);
sr.setHierarchyDomain("app.messaging.exchange");
sr.setHierarchyType("Exchange");
sr.setOpenId(new OpenId().addId("serverName", "Exchange1122"));

ExtendedAttributesData xa = new ExtendedAttributesData();
xa.addAttribute("productID", "ID12024");
xa.addAttribute("customID", "ID12333");
xa.addAttribute("Location", "floorNo", "3");
xa.addAttribute("Location", "roomNo", "15");
xa.attachTo(sr);
```

Attribut fusionné

```
hierarchyType -> Exchange
isPlaceholder -> false
openId -> <openId><id><name>servername</name><value>Exchange1122</value>
</id></openId>
hierarchyDomain -> app.messaging.exchange
XA -> <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xml>
  <attribute name="customID" category="taddm_global">ID12333</attribute>
  <attribute name="internal" category="taddm_global">ID1233</attribute>
  <attribute name="productID" category="taddm_global">ID12024</attribute>
  <attribute name="buildingNo" category="Location">North23</attribute>
  <attribute name="floorNo" category="Location">3</attribute>
  <attribute name="roomNo" category="Location">15</attribute>
</xml>
```

Les attributs de chevauchement provenant du premier objet et du deuxième objet ne sont pas tous présents avec leurs valeurs dans l'attribut fusionné et la valeur de l'attribut `productId` est issue de l'objet enregistré en dernier. Lorsque l'attribut `XA` d'un objet de modèle est mis à jour partiellement, le type de catégorie et le nom de l'attribut sont inclus dans le processus.

Mise en cache des métadonnées et validation

Les attributs étendus sont définis rarement. Pour cette raison, un cache est créé pour conserver les données. Chaque fois qu'un objet de modèle est placé dans un outil de persistance pour être enregistré, l'outil valide l'attribut `XA` par rapport aux définitions méta qui sont extraites du cache. Chaque paire type de catégorie et nom d'attribut qui est présente dans l'attribut `XA`, sans être définie dans le cache méta est supprimée de l'attribut `XA` avant que le objet de modèle ne soit conservé. Si le processus d'actualisation du cache est désactivé, les méta-informations sont obtenues à partir de la couche de persistance. Si le processus est activé, les méta-informations sont obtenues à partir de la mémoire Java.

Vous pouvez contrôler la fréquence d'actualisation du cache à l'aide de la propriété `com.ibm.cdb.ea.metaRefreshFrequency` du fichier `collation.properties`. La valeur par défaut exprimée en secondes est `20`. Pour désactiver le processus d'actualisation du cache, définissez cette propriété sur la valeur `0`.

Remarque : Lorsque vous définissez des attributs étendus, désactivez le processus d'actualisation du cache et réactivez-le une fois cette opération terminée.

Requêtes MQL avec opérateur EVAL (attributs XA et XD uniquement)

Beaucoup d'attributs CDM sont déplacés dans le contenu XML des attributs XA ou XD. Par conséquent, la syntaxe MQL prend en charge un nouvel opérateur eval, qui peut être ajouté à la clause where. L'opérateur eval permet d'interroger les éléments de configuration par les valeurs des attributs étendus ou des instances étendues.

Remarque : Tous les exemples de requêtes MQL contiennent des guillemets avec caractères d'échappement (\"*value*\") car on suppose que les requêtes sont exécutées de la manière suivante :

```
./api.sh -u username -p password find "MQL query"
```

Par exemple, la requête suivante a été exécutée pour trouver un système informatique avec l'attribut productID défini sur 'prod1' :

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La requête équivalente suivante utilise l'opérateur eval :

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

L'opérateur eval peut être suivi par une expression XPath valide qui renvoie une valeur booléenne true ou false afin de permettre l'exécution d'un filtrage SQL correct par la couche de persistance.

Autres exemples

- Rechercher tous les systèmes informatiques ayant un attribut étendu défini sur la valeur val :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```
  - SQL :

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```
- Rechercher un système informatique ayant l'attribut étendu attr2, dont la catégorie est définie sur Autre et la valeur sur two :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```
  - SQL :

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
     and and @category="Other" and text()="two"]' passing compsys.xa_x as
     "c")
    ```

Création d'attribut étendu dans le portail de gestion de domaine

Vous pouvez définir des attributs étendus pour un composant particulier dans le portail de gestion de domaine.

Procédure

Pour créer un attribut étendu, procédez comme suit dans le portail de gestion de domaine.

Remarque : Vous pouvez également définir des attributs étendus en exécutant le programme de chargement en bloc.

1. Dans la barre de menus, cliquez sur **Editer** > **Attributs étendus**. La fenêtre Définir des attributs étendus s'affiche.
2. Dans le menu **Type de composant**, sélectionnez le type de composant pour lequel vous souhaitez créer un attribut étendu.
3. Cliquez sur **Nouveau**. La fenêtre Créer un attribut étendu s'affiche.
4. Dans la zone **Nom d'attribut étendu**, entrez le nom de l'attribut étendu.
5. Dans le menu **Type d'attribut étendu**, sélectionnez le type de l'attribut étendu à créer.
6. Dans le menu **Catégorie d'attribut étendu**, indiquez la catégorie de l'attribut étendu.
7. Cliquez sur **OK**.
8. Dans la fenêtre Définir des attributs étendus, cliquez sur **OK**.
   Voir également la rubrique *Fenêtre Définir des attributs étendus* dans le *Guide d'utilisation* de TADDM.

Création des attributs étendus dans les fichiers

Vous pouvez définir des attributs étendus pour un composant particulier dans les fichiers du répertoire `$COLLATION_HOME/dist/etc/templates`.

Procédure

1. Vérifiez que les fichiers modèle du répertoire `$COLLATION_HOME/dist/etc/templates/commands` contiennent uniquement `SCRIPT:etc/templates/commands/extension-scripts/ea_sensor_1.0.py`.
2. Sur le serveur que vous souhaitez reconnaître, créez un fichier de configuration `/tmp/`*`nom_fichier`*`.conf` qui contient des paires d'attributs à définir ainsi que leurs valeurs. Par exemple :

   ```
   PremierAttribut = Première valeur
   DeuxièmeAttribut = Deuxième valeur
   ```

   Répertoriez ces attributs dans le fichier `extended_attributes.properties`. Voir l'étape suivante.
3. Dans le répertoire `$COLLATION_HOME/etc/templates`, créez le fichier `extended_attributes.properties`. Il doit contenir les attributs à collecter. Dans le fichier `extended_attributes.properties.example`, vous pouvez trouver la structure de ce fichier. Exemple d'entrée :

   ```
   Linux.FileGEN.1.Key="/tmp/attribute.conf"
   Linux.FileGEN.1.Attributes="monAttribut"
   ```

   où :
   - `/tmp/attribute.conf` est l'emplacement du fichier dans lequel vous spécifiez l'attribut que voulez définir (`/tmp/`*`nom_fichier`*`.conf`).
   - `monAttribut` est le nom de l'attribut que vous voulez définir et qui est spécifié dans le fichier `/tmp/attribute.conf`.
4. Dans la console de gestion de reconnaissance, dans la fenêtre Systèmes informatiques, définissez la valeur **Enable** à *true*.

Que faire ensuite

Avant d'exécuter une reconnaissance, tous les attributs à collecter doivent également être créés dans le portail de gestion de domaine. Si ce n'est pas le cas, ils sont ignorés pendant la reconnaissance.

Important : Pendant la reconnaissance, les noms des attributs que vous définissez dans les fichiers sont changés. Le préfixe `FileGEN` est ajouté au début de chacun des noms. Par exemple, si le nom de votre attribut est `monAttribut`, il est changé pour `FileGEN_monAttribut`. En conséquence, lorsque vous créez des attributs dans le portail de gestion de domaine, vous devez fournir des noms avec le préfixe `FileGEN`. Les attributs créés dans le fichier et dans le portail de gestion de domaine doivent avoir les mêmes noms. Si vous fournissez des noms sans le préfixe, les attributs sont ignorés.

Suppression des attributs étendus

Vous pouvez supprimer des attributs étendus existants.

Procédure

Pour supprimer un attribut étendu, procédez comme suit dans le portail de gestion de domaine :

1. Dans la barre de menus, cliquez sur **Editer** > **Attributs étendus**. La fenêtre Définir des attributs étendus s'affiche.
2. Dans le menu **Type de composant**, sélectionnez le type de composant pour lequel vous souhaitez supprimer un attribut étendu.
3. Sélectionnez un attribut étendu existant.
4. Cliquez sur **Supprimer**.
5. Cliquez sur **OK**. L'attribut étendu est supprimé.

Définition automatique des attributs étendus pour une API Java publique et un programme de chargement en bloc

Puisque la définition manuelle d'attributs étendus demande du temps, vous pouvez activer la définition automatique des attributs étendus. Les attributs étendus ne peuvent être utilisés dans le portail de gestion de données que pour corriger et régler des définitions d'attributs étendus.

Une API Java et un programme de chargement en bloc externes permettent au gestionnaire de stockage de TADDM de vérifier s'il est possible de trouver des attributs étendus attachés à un élément de configuration. Si de tels attributs sont trouvés, ceux qui n'ont pas été définis comme type `String` peuvent être définis automatiquement. Grâce à ce mécanisme, il est préférable d'utiliser le livre ou un script Jython pour stocker les données avec une définition automatique forcée, et d'accéder ensuite au portail de gestion de données pour vérifier les définitions, et ajuster les types d'attribut et catégories.

Pour activer une définition automatique des attributs étendus en utilisant une API Java externe, utilisez la méthode `setAutoDefineEAs` de l'objet `ApiSession`, qui est définie à `true`, comme dans l'exemple suivant :

```
conn = ApiFactory.getInstance().getApiConnection("localhost", -1, None,
Boolean(0))
sess = ApiFactory.getInstance().getSession(conn, "administrator", "collation",
ApiSession.DEFAULT_VERSION)
api = sess.createCMDBApi()

  print "Activation d'une fonction API de définition automatique pour des attributs étendus"
sess.setAutoDefineEAs(Boolean(1))
```

Pour permettre la définition automatique des attributs étendus en utilisant un programme de chargement en bloc, utilisez l'attribut `-loadEAMeta` ou définissez la propriété `com.ibm.cdb.bulk.loadeametaflag=true` dans le fichier `collation.properties`. Dans ce mode, le programme de chargement en bloc

n'enregistre aucun élément de configuration. En revanche, il extrait les informations relatives à tous les attributs étendus qui sont chargés, et utilisez ces attributs étendus pour créer des définitions correctes. Il en résulte que des objets `UserDataMeta` corrects sont créés dans TADDM.

Remarque : Vous pouvez utiliser l'option `-g` lorsque vous exécutez le programme de chargement en bloc pour accélérer le processus.

Définition automatique des attributs étendus pour les détecteurs

Fix Pack 2

Vous pouvez définir des attributs étendus pour un détecteur et les utiliser pour stocker automatiquement les données du détecteur.

Pour définir des attributs étendus pour les détecteurs, créez un manuel IdML contenant les définitions des attributs étendus. Le nom de ce manuel doit être `xa.xml`. Placez le fichier dans le répertoire de bundle OSGi, par exemple, `$COLLATION_HOME/osgi/plugins/com.ibm.cdb.discover.sensor.sys.examplesensor_1.0.0/xa.xml`.

Chaque fois que le service TopologyBuilder est démarré, les nouveaux manuels et les manuels modifiés sont automatiquement chargés. Au cours du processus de chargement, le programme de chargement en bloc s'exécute avec l'option `-loadEAMeta` activée, définissant ainsi automatiquement les attributs étendus. Pour plus d'informations sur le programme de chargement en bloc, voir «Définition automatique des attributs étendus pour une API Java publique et un programme de chargement en bloc», à la page 29.

Pour afficher les attributs étendus dans l'interface utilisateur de TADDM, vous devez redémarrer le serveur TADDM.

Pour configurer le processus de chargement en bloc des manuels contenant les attributs étendus, vous pouvez utiliser le fichier `$COLLATION_HOME/etc/bulkload.properties` ; il stocke la configuration du programme de chargement en bloc.

Exemple d'utilisation

Le fichier `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml` contient les définitions des attributs étendus utilisées par l'exemple FoobarSensor. Au premier démarrage du serveur, après que le service TopologyBuilder a été initialisé, TADDM charge le manuel `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml`. Puis, la somme CRC32 du fichier `xa.xml` est calculée et stockée dans le fichier `com.ibm.cdb.discover.sensor.sys.foobarsensor_1.0.0/xa.xml.crc`. La somme CRC32 est calculée dès que le fichier `xa.xml` est modifié. Les manuels sont chargés par le service TopologyBuilder uniquement lorsque la somme de vérification change.

Exemple de structure du fichier `xa.xml`

```
<idml:idml xmlns:idml="http://www.ibm.com/xmlns/swg/idml" xmlns:cdm=
"http://www.ibm.com/xmlns/swg/cdm" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/swg/idml idml.xsd">
 <idml:source IdMLSchemaVersion="0.8">
  <cdm:process.ManagementSoftwareSystem CDMSchemaVersion="2.8">
   <cdm:MSSName>vos données</cdm:MSSName>
   <cdm:Hostname>vos données</cdm:Hostname>
   <cdm:ManufacturerName>vos données</cdm:ManufacturerName>
   <cdm:ProductName>vos données</cdm:ProductName>
   <cdm:ProductVersion>vos données</cdm:ProductVersion>
   <cdm:Label>vos données</cdm:Label>
```

```
 </cdm:process.ManagementSoftwareSystem>
</idml:source>

<!-- Operation... -->
<idml:operationSet opid="1">
 <idml:create timestamp="2012-07-12T05:10:58Z">
 <cdm:CDM-ER-Specification>
  <cdm:sys.ComputerSystem id="CS" sourceToken="CS">
  <cdm:extension>
      <cdm:extattr name="sensor_foobar_xa1" category="_internal"></cdm:extattr>
  </cdm:extension>
  </cdm:sys.ComputerSystem>
 </cdm:CDM-ER-Specification>
 </idml:create>
</idml:operationSet>
</idml:idml>
```

Instances étendues

Dans l'ancien Common Data Model, les blocs de données volumineux devaient être divisés en paires clé-valeur et stockés en tant qu'attributs étendus. Désormais, vous pouvez utiliser des instances étendues pour stocker des contenus volumineux, ce qui vous évite de créer un grand nombre d'attributs étendus. Vous pouvez stocker les objets de bas niveau comme des instances étendues et les attacher à des objets sommet, qui sont des éléments de configuration, comme des données brutes ou pré-traitées.

Type de modèle de données

La structure de l'attribut `XD` est une séquence simple d'éléments XML. Chaque type d'objet de modèle possède un attribut `XD` d'un nouveau type `ExtendedInstanceData` Java. Ce type permet l'enregistrement de tout type de contenu, comme un texte en clair (par exemple les sorties de commandes), XML, CSV ou CDATA.

Vous pouvez regrouper les instances étendues par type ; le type par défaut est `Général`.

Affichage des instances étendues

Après avoir exécuté la reconnaissance d'un détecteur pour lequel vous avez spécifié l'attribut `XD`, vous pouvez afficher les instances étendues dans le portail de gestion de données. Ouvrez la sous-fenêtre Détails d'un objet reconnu. Chaque type d'instance étendue s'affiche dans un onglet distinct.

Exemple d'utilisation dans un détecteur

L'exemple suivant illustre le type `ExtendedInstanceData` qui permet d'enregistrer un contenu brut. Les variables `lsofOutput` et `ifconfigOutput` sont des variables de chaîne qui enregistrent des sorties de commandes. Dans cette commande, les sorties ne sont pas présentes, mais elles peuvent être formatées comme un langage XML ou une notation JavaScript Object Notation (JSON).

```
SComputerSystem scs = ModelFactory.newInstance(SComputerSystem.class);
scs.setHierarchyDomain("sys.unix.linux");
scs.setHierarchyType("RedHat");
scs.setManufacturer("RedHat");
scs.setModel("Linux");
scs.setSerialNumber("as00123012");

ExtendedInstanceData xd = new ExtendedInstanceData();
xd.addInstance("lsof", lsofOutput);
xd.addInstance("ipInterfaces", ifconfigOutput);
```

```
        xd.addInstance(null, "Linux vmw009128109120 2.6.32-220.el6.x86_64 #1
SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64
GNU/Linux");
        scs.setXD(xd);
```

En conséquence, la mappe d'attributs suivante avec l'attribut XD renseignée avec le code XML correct est enregistrée. Le code XML crée des éléments pour le type d'instance avec les éléments des données brutes du même type à l'intérieur, encadré par <instance>. L'instance qui est créée avec le type null est placée dans le type par défaut <general>.

```
manufacturer -> RedHat
hierarchyType -> RedHat
XD -> <xml>
 <general>
  <instance>Linux vmw009128109120 2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13
EST 2011 x86_64 x86_64 x86_64 GNU/Linux</instance>
 </general>
 <ipInterfaces>
  <instance>eth4      Link encap:Ethernet  HWaddr 00:50:56:00:72:92
          inet addr:9.128.109.120  Bcast:9.128.109.255  Mask:255.255.255.0
          inet6 addr: fe80::250:56ff:fe00:7292/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:805298636 errors:0 dropped:0 overruns:0 frame:0
          TX packets:665767802 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:590819760949 (550.2 GiB)  TX bytes:272393336858 (253.6 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:833223633 errors:0 dropped:0 overruns:0 frame:0
          TX packets:833223633 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:594042898379 (553.2 GiB)  TX bytes:594042898379 (553.2 GiB)

  </instance>
 </ipInterfaces>
 <lsof>
  <instance>pickup    31017   postfix  mem       REG               8,2  1580928
    1066619 /usr/lib64/mysql/libmysqlclient.so.16.0.0
            pickup    31017   postfix  mem       REG                 8,2   184088
     139800 /lib64/libpcre.so.0.0.1
            pickup    31017   postfix  mem       REG                 8,2    63304
     139295 /lib64/liblber-2.4.so.2.5.6
            pickup    31017   postfix  mem       REG                 8,2   308912
     130952 /lib64/libldap-2.4.so.2.5.6
            pickup    31017   postfix  mem       REG                 8,2   156872
     130819 /lib64/ld-2.12.so
            pickup    31017   postfix    0u      CHR                 1,3      0t0
       3640 /dev/null
            pickup    31017   postfix    1u      CHR                 1,3      0t0
       3640 /dev/null
            pickup    31017   postfix    2u      CHR                 1,3      0t0
       3640 /dev/null
            pickup    31017   postfix    3r     FIFO                 0,8      0t0
      10751 pipe
            pickup    31017   postfix    5u     unix 0xffff8802350d9c80      0t0
      10659 socket
            pickup    31017   postfix    6u     FIFO                 8,2      0t0
     392456 /var/spool/postfix/public/pickup
            pickup    31017   postfix    7u     unix 0xffff8802378e8680      0t0
   49305400 socket
            pickup    31017   postfix    8u      REG                 0,9        0
       3638 anon_inode
            loop0     31473      root  cwd       DIR                 8,2     4096
```

```
          2 /
              loop0     31473      root  rtd       DIR                8,2     4096
          2 /
              loop0     31473      root  txt   unknown
           /proc/31473/exe
  </instance>
 </lsof>
</xml>
serialNumber -> as00123012
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
model -> Linux
```

L'exemple précédent utilise les méthodes `addInstance` les plus simples. Il est recommandé d'utiliser la méthode `addInstance(String type, INSTANCE_FORMAT format, boolean isVisible, String content)` qui active le contrôle de deux attributs de l'élément `<instance>`.

Le premier attribut XML est `format` ; il permet d'interpréter le contenu. `INSTANCE_FORMAT` est une énumération Java avec les valeurs possibles suivantes : `plain`, `XML`, `CSV`, `JSON`, `CDATA`.

Le deuxième attribut XML est `visible` ; il est utilisé lorsque certaines instances ne sont pas affichées dans l'interface utilisateur.

Exemple :

```
<general><instance format="plain" visible="false">Linux vmw009128109120
2.6.32-220.el6.x86_64 #1 SMP Wed Nov 9 08:03:13 EST 2011 x86_64 x86_64 x86_64
GNU/Linux</instance></general>
```

Mise à jour partielle

Un élément de configuration peut être enregistré avec les mêmes valeurs des règles de nommage à partir de différentes sources de données, et en conséquence les valeurs sont fusionnées dans un objet. Pour éviter cela, nous utilisons un mécanisme de mise à jour partielle pour fusionner deux attributs `XD` formatés XML. Par exemple, si une source est enregistrée avec objet 1 et une autre source est enregistrée avec objet 2, un élément de configuration qui détient l'attribut fusionné est créé.

Objet 1

```
                SComputerSystem xd1 = ModelFactory.newInstance
        (SComputerSystem.class);
                xd1 = ModelFactory.newInstance(SComputerSystem.class);
                xd1.setHierarchyDomain("sys.unix.linux");
                xd1.setHierarchyType("RedHat");
                xd1.setManufacturer("RedHat");
                xd1.setModel("Linux");
                xd1.setSerialNumber("as00123012xd");

                ExtendedInstanceData xdone = new ExtendedInstanceData();
                xdone.addInstance("lsof", "content from CI1");
                xdone.addInstance(null, "content from CI1");
                xd1.setXD(xdone);
```

Objet 2

```
                SComputerSystem xd2 = ModelFactory.newInstance
        (SComputerSystem.class);
                xd2 = ModelFactory.newInstance(SComputerSystem.class);
                xd2.setHierarchyDomain("sys.unix.linux");
                xd2.setHierarchyType("RedHat");
                xd2.setManufacturer("RedHat");
```

```
xd2.setModel("Linux");
xd2.setSerialNumber("as00123012xd");

ExtendedInstanceData xdtwo = new ExtendedInstanceData();
xdtwo.addInstance("ips", "content from CI2");
xdtwo.addInstance(null, "content from CI2");
xd2.setXD(xdtwo);
```

Attribut fusionné

```
manufacturer -> RedHat
hierarchyType -> RedHat
XD -> <xml>
 <general>
  <instance>content from CI2</instance>
 </general>
 <lsof>
  <instance>content from CI1</instance>
 </lsof>
 <ips>
  <instance>content from CI2</instance>
 </ips>
</xml>
serialNumber -> as00123012xd
isPlaceholder -> false
hierarchyDomain -> sys.unix.linux
model -> Linux
```

Les attributs de chevauchement provenant du premier objet et du deuxième objet ne sont pas tous présents avec leurs valeurs dans l'attribut fusionné et l'attribut `general` a les instances issues de l'objet enregistré en dernier. La mise à jour partielle pour l'attribut `XD` de chaque objet de modèle est réalisée conformément au type d'instance.

Requêtes MQL avec opérateur EVAL (attributs `XA` et `XD` uniquement)

Beaucoup d'attributs CDM sont déplacés dans le contenu XML des attributs `XA` ou `XD`. Par conséquent, la syntaxe MQL prend en charge un nouvel opérateur `eval`, qui peut être ajouté à la clause where. L'opérateur `eval` permet d'interroger les éléments de configuration par les valeurs des attributs étendus ou des instances étendues.

Remarque : Tous les exemples de requêtes MQL contiennent des guillemets avec caractères d'échappement (\"*value*\") car on suppose que les requêtes sont exécutées de la manière suivante :

```
./api.sh -u username -p password find "MQL query"
```

Par exemple, la requête suivante a été exécutée pour trouver un système informatique avec l'attribut `productID` défini sur `'prod1'` :

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La requête équivalente suivante utilise l'opérateur `eval` :

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

L'opérateur `eval` peut être suivi par une expression XPath valide qui renvoie une valeur booléenne true ou false afin de permettre l'exécution d'un filtrage SQL correct par la couche de persistance.

Autres exemples

- Rechercher tous les systèmes informatiques ayant un attribut étendu défini sur la valeur `val` :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```

  - SQL :

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```

- Rechercher un système informatique ayant l'attribut étendu `attr2`, dont la catégorie est définie sur `Autre` et la valeur sur `two` :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```

  - SQL :

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
     and and @category="Other" and text()="two"]' passing compsys.xa_x as
     "c")
    ```

Extension de la portée de reconnaissance des détecteurs à l'aide du Simplified Model

Les nouvelles fonctions du Simplified Model vous permettent d'étendre facilement la portée de reconnaissance d'un détecteur. La procédure suivante montre un exemple de configuration que vous pouvez utiliser comme base pour personnaliser vos propres détecteurs.

Pourquoi et quand exécuter cette tâche

La procédure suivante décrit la configuration d'un détecteur simple basé sur un script pour la base de données MongoDB, lorsque le serveur MongoDB est installé.

Prérequis

Si le détecteur doit stocker des attributs étendus, vous devez d'abord définir ces attributs pour un type de modèle de données particulier. Pour plus d'informations, voir «Création d'attribut étendu dans le portail de gestion de domaine», à la page 27.

Pour cette procédure, deux attributs étendus ont été créés avec les paramètres suivants :

- Type `SDeployableComponent` :
  - Nom d'attribut étendu - hasDatabaseIndexDef
  - Type d'attribut étendu - Booléen
  - Catégorie d'attribut étendu - Général
- Type `SSoftwareServer` :
  - Nom d'attribut étendu - DatabaseCnt
  - Type d'attribut étendu - Entier
  - Catégorie d'attribut étendu - Markers

Remarque : La procédure suivante ne décrit pas les nouvelles fonctions du modèle simplifié. Elle montre plutôt comment utiliser ces fonctions. Pour plus d'informations, voir les rubriques appropriées dans la section «Simplified Model», à la page 19 de la documentation TADDM.

Procédure

1. Créez un modèle de serveur personnalisé à l'aide de la sous-fenêtre Serveurs personnalisés de la Console de gestion de reconnaissance. La sous-fenêtre Serveurs personnalisés contient des modèles de détecteurs qui peuvent être exécutés par le détecteur de serveur générique ou comme détecteurs de serveur d'applications personnalisé.
   Affichez les détails pour le serveur MongoDB. Celui-ci est activé. Il contient des objets de type AppServer. Dans la section Identification des critères, indiquez un nom de fichier. Lorsque le détecteur de serveur générique détecte un processus contenant ce nom de fichier, il exécute le détecteur personnalisé. Dans ce cas, le nom de fichier est `mongod.exe`.

   Vous pouvez maintenant exécuter une reconnaissance, mais seuls les objets RuntimeProcess seront reconnus.
2. Configurez l'extension pour le détecteur modèle.
   a. Ouvrez le répertoire principal TADDM et accédez au répertoire `dist/etc/templates/commands`. Créez un fichier de commandes avec le même nom que le modèle défini dans Serveurs personnalisés (dans ce cas, `MondoDB`). Le contenu de ce fichier est exécuté lorsque ce détecteur personnalisé est lancé.
      Le fichier `MongoDB` contient la ligne suivante :

      ```
      SCRIPT[com.ibm.cdb.core.jython253_2.5.3]:etc/templates/commands/
      extension-scripts/mongodb.py
      ```

      Cela signifie que le script `mongodb.py` est exécuté lorsque le détecteur est lancé.
   b. Configurez le script `mongodb.py`. Les sections suivantes fournissent une brève description des principaux éléments de ce script.

      Début par défaut du détecteur

      ```
      (os_handle, result, appserver, seed, log, env) = sensorhelper.init
      (targets)
      ```

      Cette section décrit les paramètres transmis par une plateforme appelée `targets` :

      - `os_handle` - un objet de système d'exploitation.
      - `result` - le résultat auquel les objets sont associés.
      - `appserver` - l'objet de serveur d'applications représentant les processus d'exécution qui correspondent à la condition du modèle.
      - `seed` - l'objet de départ qui déclenche le détecteur.
      - `log` - journal d'événements.
      - `env` - paramètres d'environnement.

      Création de l'objet principal - définition d'attributs de classe et de hiérarchie

      ```
      mdbserver = ModelFactory.newInstance(Class.forName('com.collation.
      platform.model.topology.simple.SSoftwareServer'))
      mdbserver.setHierarchyDomain('app.db.mongodb')
      mdbserver.setHierarchyType('MongoDBServer')
      ```

      Dans cette section, le nouveau type de classe `SSoftwareServer` est utilisé. De nouveaux attributs sont également utilisés, qui placent un élément de configuration représenté par cet objet sur les mappes de domaines. Ces attributs fournissent les informations requises sur le détecteur.

- `HierarchyDomain` - indique les niveaux de domaines. Dans ce cas, dans la valeur `app.db.mongodb`, `app` correspond à un serveur d'applications, `db` correspond au serveur de base de données et `mongodb` correspond à une base de données spécifique.
- `HierarchyType` - indique le type de l'objet. Dans ce cas, puisque la valeur `mongodb` indiquée dans l'attribut `HierarchyDomain` représente un serveur de base de données, l'attribut `HierarchyType` est défini sur `MongoDBServer`.

`OpenId` - un nouvel attribut de règle de nommage générique

```
print 'Primary IP : ' + seed.getPrimaryIpAddress().
getStringNotation()
print 'Port       : ' + str(seed.getPort())
id = OpenId().addId('IP' , seed.getPrimaryIpAddress().
getStringNotation()).addId('port' , str(seed.getPort()))
ID = OpenId.generateDisplayName(id)
mdbserver.setInstanceID(ID)
mdbserver.setOpenId(id)
mdbserver.setHostComputerSystem(os_handle.getComputerSystem())
```

Vous devez nommer l'objet que vous avez créé. Pour ce faire, vous devez définir une règle de nommage et indiquer les attributs qui fournissent des valeurs uniques. Pour les serveurs, la règle de nommage est généralement basée sur deux attributs. Il s'agit du point principal d'accès au service créé à partir de l'adresse IP principale du serveur et d'un port sur lequel ce service écoute.

Le nouvel attribut `OpenId` est un encapsuleur pour un attribut de chaîne, qui rassemble toutes les valeurs utilisées pour le calcul des identificateurs globaux uniques. Ce nouvel attribut place toujours les valeurs dans le même ordre. La méthode `addId` ajoute des objets et les définit en tant que valeur de l'attribut `OpenId`.

Ce détecteur possède également d'autres attributs prédéfinis, tels que l'ID d'instance et le système informatique hôte, qui permettent d'associer le serveur à un système informatique transmis par une plateforme en tant qu'attribut d'objet de système d'exploitation.

Définition de l'objet de résultat

```
print 'Setting mongodb server'
result.setServer(mdbserver)
```

Définissez l'objet de résultat à l'aide du serveur MongoDB. Si le détecteur échoue lors de la reconnaissance, ces informations sont conservées au minimum.

Fonction d'extraction de bases de données

```
def extractDatabases(mgbserver, dbListLines):
  databases = list([])
  dbDir = None
  currentDatabase = None
  XD = None
  XA = None

  for dbLine in dbListLines:
    print 'Processing line : ' + dbLine
    if "DATABASE" in dbLine :
      print 'Found a database ' + dbLine
      #Création de fichiers physiques
      print 'Attach as files'
      dbLineTokens = dbLine.split(" ")
      dbName = dbLineTokens[2].strip()
```

```
      dbDir = dbLineTokens[4].strip()
      print "Database name " + dbName
      print "Database directory " + dbDir

      #Création de composants déployables
      print "Create deployable component for database"
      database = ModelFactory.newInstance(Class.forName
('com.collation.platform.model.topology.simple.
SDeployableComponent'))
      database.setHierarchyDomain('app.db.mongodb')
      database.setHierarchyType('Database')
      database.setOpenId(OpenId().addId("databaseName", dbName))
      database.setName(dbName)
      database.setSoftwareServer(mgbserver)
      databases.append(database)
      currentDatabase = database
      XA = ExtendedAttributesData()
      XA.attachTo(currentDatabase)
      XD = ExtendedInstanceData()
      currentDatabase.setXD(currentXD)

    elif dbDir is not None and dbDir in dbLine and "Metadata for"
in dbLine:
      print "Found file " + dbLine + " to read for XD"
      databaseConfFileLineTokens = dbLine.split(" ")
      fileType = databaseConfFileLineTokens[3]
      fileName = databaseConfFileLineTokens[5]
      print "Instance type " + fileType
      print "Files content to read " + fileName

      if not fileName.endswith("bson"):
        cnfFileContent = readFileContent(fileName)
        print "Read content : " + cnfFileContent
        XD.addInstance(fileType,
ExtendedInstanceData.INSTANCE_FORMAT.json, 1, cnfFileContent)
        print "Added instance"

        if "index" in cnfFileContent :
          XA.addAttribute("hasDatabaseIndexDef", "true")
          xml = XA.toXML()
          print "Add XA marker for database " + dbName +
" with XML " + xml

  print "Attach files and deployable components"
  mgbserver.setDeployedComponents(tuple(databases))
  XA = ExtendedAttributesData()
  XA.addAttribute("Markers", DatabasesCnt", str(len(databases)))
  XA.attachTo(mgbserver)
  print "Add XA marker for server with xml " + xml
```

Cette fonction utilise deux conteneurs pour les données. L'un contient les attributs étendus et l'autre contient les instances étendues.

Dans la section `#Création de composants déployables`, le détecteur crée un objet pour la base de données reconnue en utilisant le type de classe `SDeployableComponent`. `HierarchyDomain` est défini sur `app.db.mongodb` car il s'agit du même domaine, mais l'attribut `HierarchyType` est défini sur `Database`. Puisque la règle de nommage est basée sur le nom de la base de données, l'attribut `openId` est défini sur `databaseName`. Un nom d'attribut prédéfini est défini (`database.setName(dbName)`) et le composant est associé (`databases.append(database)`) à un serveur à l'aide de l'attribut SoftwareServer (`database.setSoftwareServer(mgbserver)`).

Les éléments essentiels de cette section sont les deux conteneurs de données. Ces conteneurs sont créés pour la nouvelle base de données. L'attribut `XA` contient l'objet `ExtendedAttributesData`, qui stocke les attributs étendus et l'attribut `XD` contient l'objet `ExtendedInstanceData`, qui stocke les blocs de données volumineux.

`ExtendedAttributesData`

En utilisant l'attribut `XA = ExtendedAttributesData()`, le détecteur crée un objet `ExtendedAttributesData` et l'ajoute à un élément de configuration. En utilisant la méthode `XA.addAttribute`, il ajoute les attributs à cet objet (le nom et la valeur). Par exemple, dans l'attribut étendu pour l'objet de serveur, le détecteur crée l'objet `ExtendedAttributesData` et y ajoute les attributs (`DatabaseCnt` et la catégorie `Markers`). Pour convertir l'attribut vers un format pouvant être stocké, le détecteur appelle la méthode `toXML`.

`ExtendedInstanceData`

En utilisant l'attribut `XD = ExtendedInstanceData()`, le détecteur crée un objet `ExtendedInstanceData`. Dans ce détecteur, l'attribut `XD` est utilisé pour stocker le contenu des fichiers qui représentent la structure interne de la base de données. Le contenu de la base de données est lu par la fonction `readFileContent`, qui est définie comme suit :

```
def readFileContent(fileName):
  print "Open file to read : " + fileName
  f = open(fileName, 'r')
  content = ""
  for line in f:
    content = content + line + "\n"
  return content
```

La méthode suivante est utilisée pour stocker les données :

```
XD.addInstance(fileType, ExtendedInstanceData.
INSTANCE_FORMAT.json, 1, cnfFileContent)
```

Dans cet exemple, la méthode `addInstance` contient les paramètres suivants :

- `fileType` - indique le type de l'instance.
- `ExtendedInstanceData.INSTANCE_FORMAT.json` - indique le format de l'instance.
- `1` - correspond à la valeur true et indique que le contenu est visible dans la sous-fenêtre Détails
- `cnfFileContent` - indique le contenu de l'instance.

Que faire ensuite

- Vous pouvez exécuter la reconnaissance de MongoDB avec la nouvelle portée.
- Vous pouvez afficher les résultats de la reconnaissance dans la sous-fenêtre Récapitulatif de l'inventaire du portail de gestion de données. Vous pouvez également parcourir des dossiers génériques dans la sous-fenêtre Composants reconnus et afficher les détails des objets reconnus.
- Vous pouvez créer des applications métier pour l'objet que vous stockez et afficher la topologie.
- Vous pouvez voir plus d'exemples du script Jython.

Autres exemples de script Jython

Pour reconnaître des attributs étendus que vous avez définis pour la classe ComputerSystem (SComputerSystem) ou AppServer (SSoftwareServer), reportez-vous aux exemples de script Jython suivants.

Remarque : Si vous modifiez le script existant, par exemple, l'exemple de script `ea_sensor_1.0.py`, écrasez la partie principale du fichier. En outre, la section relative aux importations de bibliothèque standard doit inclure la ligne `from com.collation.platform.model.util.ea import ExtendedAttributesData`, comme dans l'exemple ci-dessous :

```
import sys
import java

from java.lang import System
from com.collation.platform.model.util.ea import ExtendedAttributesData
```

Script Jython utilisé pour étendre un modèle de système informatique, section principale

```
try:
 (os_handle, result, computerSystem, seed, log) = sensorhelper.
init(targets)

 patchVersionOut = sensorhelper.executeCommand("tail -1 /etc/
patch_status | cut –f2 –d,")
 patchDateOut = sensorhelper.executeCommand("tail -1 /etc/patch_
status | cut –f3 –d,")

 if patchVersionOut != None:
  XA = ExtendedAttributesData()
  XA.addAttribute("patchVersion", patchVersionOut)
  XA.addAttribute("patchDate", patchDateOut)
               XA.attachTo(computerSystem)
 else:
  log.info("patchVersion command return no output")
except:
 LogError("unexpected exception getting patchVersion
information")
```

Script Jython utilisé pour étendre un modèle de serveur personnalisé, section principale

```
try:
 (os_handle, result, appServer, seed, log, env) = sensorhelper.
init(targets)

 patchVersionOut = sensorhelper.executeCommand("tail -1 /etc/
patch_status | cut –f2 –d,")
 patchDateOut = sensorhelper.executeCommand("tail -1 /etc/patch_
status | cut –f3 –d,")

 if patchVersionOut != None:
  XA = ExtendedAttributesData()
  XA.addAttribute("patchVersion", patchVersionOut)
  XA.addAttribute("patchDate", patchDateOut)
  XA.attachTo(appServer)
 else:
  log.info("patchVersion command return no output")
except:
 LogError("unexpected exception getting patchVersion
information")
```

Présentation de l'API TADDM

Toutes les données de reconnaissance qui sont affichées via le portail de gestion de données sont accessibles depuis les interfaces de programme d'application (API) TADDM. Cette rubrique décrit les principales API TADDM : l'API Java, l'API SOAP, l'API REST et l'API de ligne de commande.

Présentation de l'interface de programme d'application

Vous pouvez accéder aux données de reconnaissance en utilisant des types caractéristiques de l'interface de programme d'application (API.)

Il est possible d'accéder à toutes les données de reconnaissance en utilisant les types d'API suivants :

Interface de programme d'application Java
Interface de programme d'application TADDM complète, permettant le développement et l'intégration d'applications Java.

API SOAP
Présente les éléments de l'interface de programme d'application TADDM comme un service Web de protocole SOAP (SOAP).

Interface de programme d'application REST
Présente les éléments de l'interface de programme d'applications TADDM comme un service Web RESTful.

Interface de programme d'application de l'interface de ligne de commande
Fournit un encapsuleur autour de l'API Java afin de permettre l'accès depuis la ligne de commande pour le script, la personnalisation simple et la planification.

Présentation du schéma XML

Le schéma XML de la base de données TADDM écrase la structure hiérarchique du Common Data Model (CDM) dans un document XML, avec la plupart des objets contenus intégrés dans le document.

Les méthodes d'interface de programme d'application TADDM renvoient un document XML contenant une liste d'objets spécifiés par la requête en langage XML, le cas échéant. Ce document est plus volumineux que les données d'origine, mais plus simple à explorer à l'aide d'outils tels que XQuery ou XPath.

Le logiciel SDK TADDM représente les dépendances entre les objets à l'aide d'objets de dépendances indépendants qui connectent les fournisseurs aux services dépendants à l'aide d'ID objet.

Lorsque vous formatez des documents XML en vue d'une utilisation avec le logiciel SDK, gardez les points suivants à l'esprit :

- Le langage XML est un modèle hiérarchique et n'autorise pas les cycles.
- Le paramètre nomX_propriété est un objet de modèle.
- Le paramètre nom_classe_recherchée_abrégée désigne le nom de la classe recherchée et non pas le nom de la classe actuelle.
- Les paramètres xsi:type et GUID sont des attributs XML et ne sont pas représentés comme des éléments séparés.
- Grappe : quand l'élément fait partie d'une grappe, N correspond à son index.

Le tableau suivant décrit la structure d'un document XML :

Tableau 6. Structure du document XML

XML	Description
`<?xml version="1.0" encoding="UTF-8"?>`	En-tête

`<results`
`xmlns="urn:www-collation-com:1.0"`
`xmlns:coll="urn:www-collation-com:1.0"`
`xmlns:xsi="http://www.w3.org/2001/`
`XMLSchemainstance"`
`xsi:schemaLocation=`
`"urn:www-collation-com:1.0`
`urn:www-collation-com:1.0/results.xsd">`
	Noeud de résultats de niveau supérieur, y compris la spécification de l'espace de nom XML

`<abbreviated_searched_class_name`
`xsi:type="full class name"`
`array="N"`
`LINK="guid"`
`lastModified="Last Modified Time in ms"`
`guid="unique model object id">`
	Attributs de classe, tels que la date de la dernière modification et l'ID objet unique

`<property_name1>"text value"`
`</property_name1>`
`...`
`<property_nameX`
`xsi:type="full class name"`
`guid="unique model object id">`
`<property_name1>"text`
`value"</property_name1>`
`...`
`<property_nameN>"text`
`value"</property_nameN>`
`<property_nameX/>`
`...`
`<property_nameN>"text value"`
`</property_nameN>`
`<property_nameQ/>`
	Valeurs d'attribut et objets imbriqués

`</abbreviated_searched_class_name>`	Fin des valeurs
`</results>`	Fin des résultats

Présentation du format JSON

L'interface de programme d'application REST utilise le format JSON pour renvoyer des données représentant des objets de modèle. Vous pouvez demander une sortie JSON en spécifiant le paramètre `feed=json` sur un appel d'interface de programme d'application REST.

Le tableau suivant décrit la structure du format JSON utilisé pour représenter des objets de modèle.

Elément JSON	Description
`[`	Démarre un tableau d'objets de modèle.
`{`	Démarre un objet de modèle, qui peut contenir zéro ou n objets de modèle.
`"`*`nom`*`":`*`valeur`*`,"`*`nom`*`":`*`valeur`	Une ou plusieurs paires valeur-nom, séparées par une virgule.

Elément JSON	Description
"_class":"*nom_classe*"	Paire valeur-nom requise contenant le nom de l'objet de modèle. Le nom de classe de l'objet de modèle peut se présenter sous deux formes : • Nom abrégé (par exemple, ComputerSystem) • Nom qualifié complet (par exemple, *com.collation.platform.model.topology.sys.ComputerSystem* Si vous spécifiez `longClassName=true` lors d'une requête REST, toutes les valeurs renvoyées pour `_class` contiennent le nom du modèle complet. Sinon, le nom abrégé est renvoyé, sauf s'il n'est pas unique (dans ce cas, le nom complet est renvoyé).
}	Termine un objet de modèle.
]	Termine un tableau d'objets de modèle.

L'exemple suivant présente la sortie JSON d'une requête ComputerSystem query à une profondeur de 1 :

```
[{
"displayName":"esx3-vm16-rhes4",
"devices":[{"_class":"DiskDrive","guid":"2A2827686EB03465A955DE54BD3F6AB5"},
{"_class":"DiskDrive","guid":"D7DAF9DCD1E7347684A0D02E36E212DC"}],
"lastModifiedBy":"system",
"l2Interfaces": [{"_class":"L2Interface","guid":"FA048919AA953BA5A09580496017A776"},
{"_class":"L2Interface","guid":"297B125690B33B778C347E12CFC62689"}],
"createdBy":"system",
"_class":"LinuxUnitaryComputerSystem",
"controllers":[{"_class":"Controller","guid":"7B72D3B5448D30388F9D9497EA8F970D"},
{"_class":"Controller","guid":"B619ABB8B8343C1FAB5BF87AD425559E"}],
"guid":"C2D379A936433258BABBF682A8E71A82",
"CPUSpeed":3191000000,
"fqdn":"esx3-vm16-rhes4",
"contextIp":"9.43.73.87",
"OSInstalled":[{"_class":"Linux","guid":"04BFCBCD2A1733258F5C95CD281D91AF"}],
"memorySize":3988783104,
"ipInterfaces":[{"_class":"IpInterface","guid":"9CAA8E0197333BAD924EA3CCB1860920"},
{"_class":"IpInterface","guid":"C2E6D21CF24435EABCB8AA8136BB9F1B"}],
"signature":"9.43.73.87(000C29A467A9)",
"systemId":"2b095749",
"bidiFlag":3,
"name":"esx3-vm16-rhes4",
"OSRunning":{"_class":"Linux","guid":"04BFCBCD2A1733258F5C95CD281D91AF"},
"CPUType":"Intel(R) Xeon(TM) MV",
"type":"ComputerSystem",
"numCPUs":1,
"architecture":"i686",
"fileSystems":[{"_class":"UnixFileSystem","guid":"CDA94FB8C84B300ABA2A42E1EFEE6234"},
{"_class":"NFSFileSystem","guid":"6300742848BA39478EAEE4FB4709DF7A"}],
"lastModifiedTime":1225806427541
}]
```

Présentation du langage MQL

La commande **rechercher()** de l'interface de programme d'application TADDM accepte une chaîne de requête, indiquée à l'aide du langage MQL (Model Query Language). Le MQL agit comme un filtre pour limiter les objets sélectionnés.

Le langage MQL utilise une syntaxe semblable à celle du langage SQL qui permet de spécifier la classe d'objet du modèle ou d'autres sources de données, leurs attributs, ainsi qu'une expression de filtrage.

La syntaxe d'une requête MQL se présente comme suit :

```
SELECT liste-attribut FROM sources-de-données [ expression WHERE ]
```

Le tableau 7 décrit les éléments d'une requête MQL. Le MQL n'est pas sensible à la casse.

Tableau 7. Eléments d'une requête MQL

Elément	Description
liste d'attributs	La valeur * ou une liste séparée par des virgules d'attributs de la classe ModelObject source. Il est également possible de spécifier des attributs intégrés. Un nom d'attribut commence toujours par une lettre minuscule, sauf si la première et la deuxième lettres sont toutes deux des majuscules. Les lettres suivantes du nom d'attribut peuvent correspondre à des majuscules ou des minuscules. Voici quelques exemples de noms d'attribut : • displayName • fqdn • OSRunning
sources de données	Liste séparée par des virgules de noms de classe d'objets modèles. Ces classes doivent être permanentes puisque vous ne pouvez pas lancer de requête sur des objets non permanents.

expression	Expression de filtrage, exprimée à l'aide du format suivant :
	`member-name OP expression [ ... ]`
	
	où :
	• Member-name est un attribut de la source de données sélectionnée et peut inclure des membres séparés par des points (les classes de membre spécifiées dans l'expression de requête doivent être permanentes. Vous pouvez interroger les attributs de membre pour obtenir les valeurs correspondant à l'expression de requête).
	• OP est un opérateur.
	• Expression est une instruction qui renvoie une valeur.
	
	Par exemple :
	`SELECT *`
	`FROM ComputerSystem`
	`WHERE OSRunning.OSName == 'Linux'`
	
	Dans ce cas, OSRunning est un objet de système d'exploitation référencé par un système informatique (ComputerSystem), et OSName est un membre primitif.
	
	Pour obtenir une description des opérateurs et des priorités associées, voir tableau 8.

Le tableau 8 décrit la priorité de l'opérateur MQL, les valeurs les plus élevées représentant un niveau de priorité supérieur.

Tableau 8. Priorité de l'opérateur MQL

Jeton	Opérateur	Priorité
ou	opération OU	1
et	opération ET	2
instance-de	est une instance de	3

Tableau 8. Priorité de l'opérateur MQL (suite)

Jeton	Opérateur	Priorité
==	est égal à	3
!=	n'est pas égal à	3
>	supérieur à	3
>=	supérieur ou égal à	3
<	inférieur à	3
<=	inférieur ou égal à	3
débute-par	débute-par	4
finit-par	finit-par	4
est-égal-à	est égal à	4
est-différent-de	est différent de	4
est-vide	est Null	4
n'est-pas-vide	n'est pas Null	4
dans	dans	5
()	parenthèses	5
existe	contenu de grappe	5
supérieur()	fonction	5
inférieur()	fonction	5
!	non unaire	5
.	sélection par point	6
contient	contient	3
eval	eval	3

Le MQL ne prend pas en charge les fonctions ou opérateurs SELECT SQL suivants:
- GROUP BY
- HAVING
- DISTINCT
- nested SELECTs
- BETWEEN
- Aggregates

Vous pouvez spécifier l'opérateur logique AND comme 'et', AND, ou &&, et l'opérateur logique OR comme 'ou', OR, ou ||. De plus, vous devez mettre toutes les chaînes entre guillemets simples, par exemple, 'IBM'.

Jointures

Le langage MQL prend en charge les jointures internes gauches des objets de modèle, comme illustré par l'exemple suivant :

```
SELECT Db2Server.* FROM Db2Server, OracleInstance WHERE Db2Server.port == OracleInstance.port
```

Cette jointure renvoie tous les objets de modèle Db2Server dans les cas où les numéros de port de Db2Server et d'OracleInstance sont identiques. Le MQL ne prend pas en charge les combinaisons de jointures extérieures droites, extérieures gauches, complètes ou croisées.

Limitations

Sur DB2 version 9.5, les opérateurs `equals` et `not-equals` échouent lorsqu'ils sont exécutés sur des attributs du type de données CLOB dans la base de données. L'exception suivante est affichée :

```
com.ibm.db2.jcc.am.SqlSyntaxErrorException: DB2 SQL Error: SQLCODE=-418,
SQLSTATE=42610
```

Grammaire de l'instruction SELECT

L'exemple suivant montre la grammaire de l'instruction SELECT. Consultez la documentation Javadoc pour plus de détails et les dernières mises à jour.

```
statement := SELECT attribute-list [EXCLUDING attribute-list]
   FROM [ONLY] class_list { WHERE [expression |
   exists_expr] }
[ FETCH FIRST n { ROW | ROWS } [ ONLY ]] [ ORDER BY order_list ]
attribute_list := attrib {, attrib}* | *
class_list := domain_class {, domain_class }*
class := <a model object class>
exists_expr := exists( array_attrib op value {logical_op array_attrib op value}* )
expression := [ attrib op value | attrib post-op | pre-op ( attrib )
   |[ NOT ] IN ( expression [, ...] ){logical_op expression}*
value := <data value>
in_expr := [ NOT ] IN ( expression [, expression ... ])
array_attrib := <série d'attributs dont au moins l'avant-dernier
   attribute is an array >
op := != | == | > | < | >= | <= | contains | starts-with | ends-with |
   equals | not-equals | instanceof | eval
logical_op := AND | OR | && | ||
post_op := is-null | is-not-null
pre_op := lower | upper
attrib := {class .} [<an attribute of a class>{.embedded_attribute} | * ]
embedded_attribute := [<embedded attribute>{.embedded_attribute} | *]
domain_class := {domain_list} class
domain_list := domain {, domain}* :
domain := <the server from which to pull data from, default: local database>
order_list := attrib [ ASC | ASCENDING | DESC | DESCENDING ] [ , order_list ]
```

Les attributs peuvent contenir des caractères génériques. De même, tous les mots clé, tels que SELECT, FROM et WHERE sont insensibles à la casse.

Remarque : `value` peut avoir le type `attrib` quand l'opérateur de base `op` (!=|==|>|<|>=|<=) est utilisé.

Exemples

L'exemple suivant présente une requête MQL qui filtre les systèmes informatiques sous système d'exploitation Linux :

```
SELECT *
   FROM ComputerSystem
   WHERE OSRunning.OSName == 'Linux'
```

La requête suivante se sert de l'opérateur EXISTS pour demander une appartenance à une grappe, correspondant à tous les systèmes informatiques qui ont une interface en mode écoute sur ibm.com ou leur masque de réseau défini sur 255.255.255.0 :

```
SELECT *
 FROM ComputerSystem
 WHERE EXISTS (ipInterfaces.ipNetwork.name ends-with '.ibm.com'
  OR ipInterfaces.ipNetwork.netmask == '255.255.255.0')
```

La requête suivante sélectionne tous les systèmes informatiques dont l'attribut virtuel est défini sur true :

```
SELECT *
   FROM ComputerSystem
   WHERE virtual
```

La requête suivante sélectionne tous les systèmes informatiques dont l'attribut virtuel est défini sur false :

```
SELECT *
   FROM ComputerSystem
   WHERE not virtual
```

La requête suivante sélectionne tous les systèmes d'exploitation bénéficiant de l'attribut de service dont le nom contient "Sans fil". Puisque l'attribut de service installé est uniquement disponible sur les systèmes d'exploitation Windows, vous devez utiliser une jointure.

```
SELECT OSInstalled
 FROM ComputerSystem,WindowsOperatingSystem
 WHERE ComputerSystem.guid==WindowsOperatingSystem.parent.guid
 AND
 EXISTS(WindowsOperatingSystem.installedServices.displayName contains 'Wireless')
```

La requête suivante sélectionne tous les serveurs d'application (AppServers) avec l'attribut primarySAP qui a pour valeur un port spécifié :

```
SELECT primarySAP.portNumber,displayName
    FROM AppServer
    WHERE primarySAP.portNumber==9084
```

La requête suivante sélectionne tous les processus d'exécution (RuntimeProcesses) qui parmi leurs ports ont le port 1415. La requête doit utiliser l'opérateur EXISTS car l'attribut ports pour le processus d'exécution (RuntimeProcess) est un attribut de tableau.

```
SELECT ports.portNumber,displayName
   FROM RuntimeProcess
   WHERE EXISTS (ports.portNumber==1415)
```

Requêtes MQL avec l'opérateur NOT EQUAL TO (!=)

Les requêtes avec l'opérateur NOT EQUAL TO ne renvoient pas de résultats contenant un attribut défini sur NULL car un tel résultat est considéré comme "inconnu".

Exemple

On considère souvent que le nombre de résultats qui sont renvoyés depuis la commande de recherche d'interface de programme d'application suivante :

```
./api.sh -u <admin> -p <pass> find --count "select * from ComputerSystem"
```

est égal à la somme des résultats qui sont renvoyés depuis les deux commandes de recherche d'interface de programme d'application suivantes :

```
./api.sh -u <admin> -p <pass> find --count "select * from
ComputerSystem where manufacturer == 'IBM'"
```

```
./api.sh -u <admin> -p <pass> find --count "select * from
ComputerSystem where manufacturer != 'IBM'"
```

Cependant, l'attribut Fabricant est défini sur NULL, il est donc exclu des résultats renvoyés depuis la requête qui contient l'opérateur NOT EQUAL TO.

Les requêtes qui contiennent l'opérateur NOT EQUAL TO peuvent se présenter sous les formes suivantes :

```
select * from ComputerSystem where manufacturer != 'IBM'
```

```
select * from ComputerSystem where not(manufacturer == 'IBM')
```

Si vous souhaitez sélectionner tous les systèmes informatiques avec un fabricant différent d'IBM, utilisez la requête suivante :

```
select * from ComputerSystem where manufacturer != 'IBM'
or manufacturer is-null
```

Requêtes MQL avec opérateur EVAL (attributs XA et XD uniquement)

Beaucoup d'attributs CDM sont déplacés dans le contenu XML des attributs XA ou XD. Par conséquent, la syntaxe MQL prend en charge un nouvel opérateur eval, qui peut être ajouté à la clause where. L'opérateur eval permet d'interroger les éléments de configuration par les valeurs des attributs étendus ou des instances étendues.

Remarque : Tous les exemples de requêtes MQL contiennent des guillemets avec caractères d'échappement (\"*value*\") car on suppose que les requêtes sont exécutées de la manière suivante :

```
./api.sh -u username -p password find "MQL query"
```

Par exemple, la requête suivante a été exécutée pour trouver un système informatique avec l'attribut productID défini sur 'prod1' :

```
SELECT * FROM ComputerSystem WHERE productID == 'prod1'
```

La requête équivalente suivante utilise l'opérateur eval :

```
SELECT * FROM ComputerSystem WHERE XA eval '/xml
[attribute[@category=\"taddm_global\" and @name=\"productID\"]=\"prod1\"]'
```

L'opérateur eval peut être suivi par une expression XPath valide qui renvoie une valeur booléenne true ou false afin de permettre l'exécution d'un filtrage SQL correct par la couche de persistance.

Autres exemples

- Rechercher tous les systèmes informatiques ayant un attribut étendu défini sur la valeur val :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml[attribute=\"val\"]'
    ```
  - SQL :

    ```
    SELECT * FROM compsys WHERE xmlexists('$c/xml[attribute="val1"]'
    passing compsys.xa_x as "c")
    ```
- Rechercher un système informatique ayant l'attribut étendu attr2, dont la catégorie est définie sur Autre et la valeur sur two :
  - MQL :

    ```
    SELECT * FROM ComputerSystem WHERE XA eval '/xml/attribute
    [@name=\"attr2\" and @category=\"Other\" and text()=\"two\"]'
    ```
  - SQL :

```
SELECT * FROM compsys WHERE xmlexists('$c/xml/attribute[@name="attr2"
 and and @category="Other" and text()="two"]' passing compsys.xa_x as
 "c")
```

Utilisation de l'interface de programme d'application Java

L'interface de programme d'application Java permet de contrôler le processus de reconnaissance et les aspects du Common Data Model y compris l'accès aux données de modèle obtenues.

Grâce à l'interface de programme d'application Java, vous pouvez créer des applications qui permettent d'ajouter, de mettre à jour et de supprimer les objets de modèle. Vous pouvez lancer des requêtes sur des objets de modèle par nom de classe ou numéro ID objet. Cette interface vous permet également de gérer les relations entre objets, d'effectuer des comparaisons et d'examiner l'historique des changements.

Les données de modèle peuvent être filtrées sur le serveur et sont retournées au client au format XML. Vous pouvez alors effectuer des transformations et lancer d'autres requêtes sur le client, au besoin. L'interface de programme d'application Java propose également des méthodes permettant de gérer les sessions et de mettre en oeuvre des opérations liées à la sécurité.

L'interface de programme d'application Java se trouve dans la classe **`com.collation.proxy.api.client.CMDBApi`**, qui communique avec une invocation RMI ApiServer sur le serveur TADDM.

Vous devez définir la propriété com.collation.home sur `$COLLATION_HOME/sdk` dans le répertoire principal de distribution SDK. A l'aide de la ligne de commande Java, vous pouvez définir la propriété comme suit :

```
java -Dcom.collation.home=$COLLATION_HOME/sdk main_classname
```

Avant d'utiliser l'interface de programme d'application Java

Avant de créer des applications Java, vous devez vérifier que votre environnement de développement est correctement configuré.

Procédure

Pour vérifier votre environnement, procédez comme suit :

1. Vérifiez que les variables d'environnement sont correctement définies puis configurez la propriété `com.ibm.cdb.service.registry.public.port`.

   Pour plus d'informations sur la définition des variables d'environnement et des paramètres de configuration spécifiques, consultez le chapitre sur la configuration du logiciel SDK TADDM.
2. Vérifiez que le serveur TADDM est en cours d'exécution.

   Pour plus d'informations, consultez le chapitre sur la vérification de l'installation du logiciel SDK.

Présentation d'un modèle d'application Java

Ce chapitre décrit comment créer, compiler et exécuter une application Java simple.

Procédure

Pour créer le modèle d'application Java, procédez comme suit :

1. Copiez le code Java suivant dans un fichier appelé `FindXmlExample.java`.

Le code source est également disponible dans le répertoire $COLLATION_HOME/sdk/examples/java.

```
package com.collation.proxy.api.examples.java;
// package com.collation.proxy.api.examples.java;

import com.collation.proxy.api.client.ApiConnection;
import com.collation.proxy.api.client.ApiException;
import com.collation.proxy.api.client.ApiSession;
import com.collation.proxy.api.client.CMDBApi;
import com.collation.proxy.api.client.DataResultSet;
import com.ibm.cdb.api.ApiFactory;

/**
 * Exemple de commande findXML() d'interface de programme d'application CMDB simple :
 * <p>  get connection and log into api server
 * <p>  find all machines which have more than 1 CPU
 * <p>  find all Oracle instances
 */

public class FindXmlExample {

  public static void main(String[] args) {

    CMDBApi api = null;
    ApiSession sess = null;
    try {
      /*
       * Etablir une connexion avec le serveur de l'API
       * <p>    ApiConnection.getConnection(host, port,
                                          trustoreLocation, useSSL)
       */
      ApiConnection conn = ApiFactory.getInstance().getApiConnection("localhost", -1,
                                              null, false);
      /*
       * Obtenir une session
       * <p>     ApiSession.getSession(connection, username,
                                     password, version)
       */
      sess = ApiFactory.getInstance().getSession(conn, "smartoperator",
                                              "foobar",
                                              ApiSession.DEFAULT_VERSION);
      /*
       * Obtenir une instance CMDBApi
       */
      api = sess.createCMDBApi();

      System.out.println("all machines which have more than 1 CPU:");
      String query = "select * from ComputerSystem where numCPUs>1";
      /*
       * Recherchez tous les ComputerSystem ayant plusieurs unités centrales.
       * La méthode : findXml(query, depth, indent, mssGuid, permissions)
       * est dépréciée, car l'ensemble de résultats est peut-être trop volumineux
       * pour être intégré à la mémoire.  En revanche, l'utilisation de curseurs
       * est préconisée :
       */
      DataResultSet data = api.executeQuery(query, null, null);
      while (data.next()) {
          System.out.println(data.getXML(4));
      }
      data.close();
      System.out.println("\nall Oracle instances:");
      query = "select * from OracleInstance";
      data = api.executeQuery(query, null, null);
      while (data.next()) {
          System.out.println(data.getXML(4));
      }
```

```
      data.close();

    } catch (ApiException ae) {
      System.err.println("api exception:" + ae);
      ae.printStackTrace();
    } catch (Exception ex) {
      System.err.println("exception:" + ex);
      ex.printStackTrace();
    } finally {
      try {
        if (api != null) {
            api.close();
        }
        if (sess != null) {
            sess.close();
        }
      } catch (Exception ex) {
        System.err.println("exception:" + ex);
        ex.printStackTrace();
      }
    }

  }
}
```

2. Par défaut, le programme exemple établit une connexion avec un serveur TADDM sur le système hôte local. Pour vous connecter à un serveur distant, modifiez la ligne suivante :

   ```
   ApiConnection conn = ApiFactory.getInstance().getApiConnection("localhost", -1,
     null, false);
   ```

   Par exemple, pour vous connecter à un serveur nommé taddmhost.ibm.com en utilisant les ports par défaut :

   ```
   ApiConnection conn = ApiFactory.getInstance().
        getApiConnection("taddmhost.ibm.com",-1, null, false);
   ```

   Par défaut, le programme exemple crée une session avec l'ID utilisateur `smartoperator` et le mot de passe `foobar`. Remplacez ces valeurs par un ID utilisateur et un mot de passe définis sur votre serveur TADDM. Par exemple :

   ```
   sess = ApiFactory.getInstance().
         getSession(conn, "administrator", "collation",ApiSession.DEFAULT_VERSION);
   ```

3. Pour compiler le programme exemple avec les autres programmes exemples Java, procédez comme suit :
   a. Sous UNIX :
      1) Modifiez pour le répertoire `$COLLATION_HOME/sdk/examples/java`.
      2) Générez un exécutable build.sh :

         ```
         chmod +x build.sh
         ```

      3) Exécutez la commande build :

         ```
         ./build.sh
         ```

   b. Sous Windows :
      1) Modifiez pour le répertoire `%COLLATION_HOME%\sdk\examples\java`.
      2) Exécutez la commande build :

         ```
         build.bat
         ```

   Si le logiciel SDK est installé séparément du serveur TADDM, assurez-vous que **`javac`** est déjà dans le chemin et disponible.
4. Exécutez l'application Java à l'aide d'une commande semblable à celle de l'exemple suivant :

```
% java -Dcom.collation.home=$COLLATION_HOME/sdk FindXmlExample
```

Cette commande exécute le programme exemple et récupère les données XML du serveur TADDM.

Informations détaillées sur le modèle d'application Java

Cette section décrit le fonctionnement de l'exemple `FindXmlExample.java`.

```
/*
 * Etablir une connexion avec le serveur de l'API
 */
  ApiConnection conn = ApiFactory.getInstance().
     getApiConnection("localhost", -1,null, false);
```

Ce segment créé un nouvel objet ApiConnection sur le serveur d'interface de programme d'application qui est utilisé comme un indicateur permettant de gérer la session d'interface de programme d'application entre le programme client et le serveur TADDM. Les arguments sont représentés dans la liste suivante :

- L'argument `host` représente le système sur lequel le serveur TADDM est exécuté.
- Le second argument correspond au port du serveur. Une valeur de -1 indique le port par défaut tel que configuré par `com.ibm.cdb.service.registry.public.port` dans le fichier de propriétés `$COLLATION_HOME/sdk/etc/collation.properties` .
- Les troisième et quatrième arguments contrôlent l'accès SSL.

```
/*
 * Obtenir une session
 *<p> ApiSession.getSession(connection, username, password,
    version)
 */
 sess = ApiFactory.getInstance().
    getSession( conn, smartoperator, foobar,ApiSession.DEFAULT_VERSION);
/*
 * Obtenir une instance CMDBApi
 */
 api = sess.createCMDBApi();
```

Ce segment connecte l'objet CMDBApi au serveur TADDM. Si vous devez vous connecter à plusieurs serveurs, dans un scénario de centre de données réparties à grande échelle, par exemple, vous pouvez utiliser plusieurs objets CMDBApi, chacun conservant le contexte d'une instance de serveur TADDM spécifique.

```
System.out.println("all machines which have more than 1 CPU:");
  String query = "select * from ComputerSystem where numCPUs>1";
  /*
  * Recherchez tous les ComputerSystem ayant plusieurs unités centrales.
  * La méthode : findXml(query, depth, indent, mssGuid, permissions)
  * est dépréciée, car l'ensemble de résultats est peut-être trop volumineux
  * pour être intégré à la mémoire.  En revanche, l'utilisation de curseurs
  * est préconisée :
  */
            DataResultSet data = api.executeQuery(query, null, null);
```

Ce segment utilise un objet CMDBApi initialisé pour extraire des données de serveur à partir du serveur TADDM à l'aide de la méthode executeQuery. Les arguments sont décrits ci-après :

- Le premier argument désigne la requête. Pour plus d'informations, voir les éléments de requête MQL dans le chapitre de présentation du langage MQL.
- Le deuxième argument correspond à l'identificateur global unique des systèmes logiciels de gestion (MSS). La valeur NULL indique que les résultats sont renvoyés par la requête pour tous les enregistrements, quel que soit le MSS auquel ils sont associés.

- Le troisième argument représente le tableau d'autorisations. Les autorisations sont définies lors de la configuration du contrôle d'accès au niveau des données. Les autorisations indiquées ici doivent correspondre à celles utilisées pour configurer le contrôle d'accès au niveau des données. Par exemple, indiquer une autorisation de mise à jour limiterait les objets renvoyés à ceux que le demandeur est autorisé à mettre à jour selon la configuration du contrôle d'accès au niveau des données. La valeur NULL indique que tous les objets auxquels l'utilisateur a accès seront renvoyés.

Référence associée:

«Systèmes logiciels de gestion», à la page 61
Les méthodes de systèmes logiciels de gestion vous permettent de gérer les systèmes logiciels de gestion dans un Common Data Model. Les méthodes de systèmes logiciels de gestion permettent d'enregistrer et de supprimer un système logiciel de gestion dans un Common Data Model. Vous pouvez également extraire des informations sur les systèmes logiciels de gestion qui ont été enregistrés avec la base de données TADDM.

Valeurs recommandées

Ce chapitre décrit les valeurs recommandées suivantes lors de l'utilisation de l'interface de programme d'application Java :

- Optimisation de l'accès aux données
- Liens à suivre entre les objets de modèle

Optimisation de l'accès aux données

Comme avec les autres interfaces de programme d'application d'accès aux données, l'interface de programme d'application TADDM peut renvoyer de grands volumes de données pouvant potentiellement submerger les ressources système. Par conséquent, évitez d'extraire toutes les données dans des environnements volumineux. Cette méthode nécessite de fréquentes synchronisations avec la base de données TADDM pour garantir la capture de toutes les modifications.

Les options suivantes constituent différentes méthodes d'extraction des données :

- Nous vous suggérons d'extraire uniquement les éléments et les identités, sans nécessairement extraire les données de configuration détaillées. Vous pouvez ainsi limiter le volume de données qui est transféré. Lorsque vous avez besoin de la configuration détaillée d'un élément, vous pouvez effectuer un appel findChanges() ultérieurement en utilisant l'ID objet comme paramètre.
- Effectuer un accès aux données de modification progressif. Pour ce faire, vous devez utiliser le type d'appel suivant avec la méthode findChanges() :

  ```
  findChanges (java.lang.String root, java.lang.String query, int depth,
  long start, long end,int changeType)
  ```

  Les paramètres **start** et **end** indiquent une période, tandis que le paramètre **changeType** indique l'état Crée, Supprimé ou Mis à jour. Cet appel findChanges() retourne uniquement les objets correspondant au type indiqué par le paramètre **changeType** dans la période donnée. Utilisez cette méthode lorsque vous effectuez une synchronisation incrémentielle des données de topologie après un transfert complet des données de référence.
- La méthode find renvoie toutes les données en une seule fois, ce qui peut entraîner des problèmes d'utilisation de la mémoire. Pour éviter ce problème, utilisez la méthode executeQuery pour parcourir les données à l'aide des curseurs :

```
DataResultSet rs = api_.executeQuery("select * from ComputerSystem", null, null);
     while (rs.next()) {
                ...
     }
     rs.close();
```

- Utilisez la méthode findCount pour compter de manière efficace le nombre d'objets correspondant à une requête :

```
long count = api.findCount("select * from ComputerSystem", null);
```

Paramètres de mémoire étendus pour le gestionnaire de topologie

Si vous lancez des requêtes non spécifiques sur des bases de données volumineuses, des incidents liés à la mémoire risquent de survenir sur le serveur TADDM et le client de l'interface de programme d'application. Utilisez des requêtes spécifiques pour identifier la taille des ensembles de résultats et de mémoire requise.

Pour les requêtes plus génériques qui renvoient un ensemble de résultats important, il est nécessaire d'allouer davantage de mémoire. Si vous recevez un message d'erreur vous signalant une mémoire insuffisante, augmentez la mémoire disponible pour la machine virtuelle Java.

Pour augmenter la mémoire, utilisez les valeurs suivantes :

- Sur le serveur TADDM, vous pouvez augmenter la mémoire disponible en modifiant le fichier de configuration de déploiement du serveur : :
  - Serveur de domaine : `cmdb-context.xml`
  - Serveur de synchronisation : `ecmdb-context.xml`
  - Serveur de stockage : `storage-server-context.xml`
  - Serveur de reconnaissance : `discovery-server-context.xml`

  Dans le fichier approprié, localisez la propriété `jvmArgs` pour la machine virtuelle Java concernée par le message d'erreur signalant une mémoire insuffisante et augmentez la mémoire en modifiant la propriété `DTaddm.xmx64`.
- Dans le client API, augmentez la mémoire spécifiée pour l'application client, par exemple dans les fichiers `api.sh` ou `api.bat`.

Liens à suivre entre les objets de modèle

La plupart des éléments de données du Common Data Model sont autonomes. Dans de nombreux cas, les liens entre les objets de modèle, tels que LogicalDependency, sont représentés par des ID objets d'archive que l'interface de programme d'application TADDM ne suit pas automatiquement. Dans ces cas là, vous devez appliquer une autre logique dans l'application de votre client.

Récapitulatif de la méthode d'interface de programme d'application Java

Grâce à l'interface de programme d'application Java, vous pouvez créer des applications qui permettent d'ajouter, de mettre à jour et de supprimer les objets de modèle. Vous pouvez lancer des requêtes sur des objets de modèle par nom de classe ou numéro ID objet. Cette interface vous permet également de gérer les relations entre objets, d'effectuer des comparaisons et d'examiner l'historique des changements de la base de données TADDM.

L'interface de programme d'application Java peut être résumée à l'aide des catégories suivantes qui sont expliquées en détail dans les chapitres respectifs :

- Gestion des sessions
- Gestion de la reconnaissance
- Gestion du modèle
- Opérations de recherche, de mise à jour et de suppression
- Gestion des collections
- Gestion des relations
- Métadonnées
- Systèmes logiciels de gestion
- MSSObjectLink
- Historique des changements
- Présentation
- Gestion des versions
- Sécurité
- Création et gestion de listes d'accès
- Gestion des modèles d'application

Historique des changements :

Les méthodes d'historique des changements permettent de déterminer l'historique des changements dans le Common Data Model. Les méthodes d'historique des changements vous permettent d'extraire l'historique des changements des éléments gérés dans le Common Data Model. Vous pouvez également déclencher la propagation et le calcul de l'historique des changements, au besoin.

Le tableau 9 décrit les méthodes d'historique des changements que vous pouvez utiliser.

Tableau 9. Méthodes d'historique des changements

Méthode	**Description**
getChangeHistory(Guid[] Guids, long start, long end)	Renvoie l'historique des changements de la grappe spécifiée des identificateurs globaux uniques.
getChangeHistory(Guid[] Guids, long start, long end, int filterType)	Reçoit l'historique des changements de plusieurs identificateurs globaux uniques, filtrés par le paramètre filterType. Cette méthode est analogue à la méthode getChangeHistory() à laquelle filterType a été ajouté et peut être définie sur l'une des valeurs suivantes : • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangeHistory(Guid Guid, long start, long end)	Renvoie l'historique des changements de l'identificateur global unique spécifié.
getChangeHistory(String[] classNames, long start, long end)	Renvoie l'historique des changements pour les classes spécifiées.
getChangeHistoryByPersistTime(String[] classNames, long start, long end)	Renvoie l'historique des changements pour les classes persistantes basées sur le temps spécifié.

Tableau 9. Méthodes d'historique des changements (suite)

Méthode	Description
getChangeHistoryInXML(Guid[] Guids, long start, long end)	Renvoie l'historique des changements de la grappe spécifiée des ID objet.
getChangeHistoryInXML(Guid[] Guids, long start, long end, int filterType)	Reçoit l'historique des changements de plusieurs identificateurs globaux uniques, filtrés par le paramètre filterType. Cette méthode est analogue à la méthode getChangeHistory() à laquelle filterType a été ajouté et peut être définie sur l'une des valeurs suivantes : • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangeHistoryInXML(Guid Guid, long start, long end)	Renvoie l'historique des changements de l'identificateur global unique spécifié.
getPropagatedChanges(long primaryKey) **Remarque :** Cette méthode est obsolète. Utilisez la méthode getChangeHistory.	Reçoit les raisons d'origine d'un historique des changements donné, renvoyé comme une représentation XML des objets ChangeHistory qui ont engendrés le changement actuel. Par exemple, si un module de serveur Apache est modifié, les changements sont propagés au serveur Apache de niveau supérieur. Vous pouvez utiliser cette méthode pour déterminer la raison qui a déclenché le changement effectué sur le serveur Apache.
getPropagatedChangesInXML(long primaryKey) **Remarque :** Cette méthode est obsolète. Utilisez la méthode getChangeHistory.	Reçoit les raisons d'origine d'un historique des changements donné, renvoyé comme une représentation XML des objets ChangeHistory qui ont engendrés le changement actuel. Par exemple, si un module de serveur Apache est modifié, les changements sont propagés au serveur Apache de niveau supérieur. Vous pouvez utiliser cette méthode pour déterminer la raison qui a déclenché le changement effectué sur le serveur Apache.
processChanges()	Déclenche la propagation et le calcul de l'historique des changements depuis la dernière reconnaissance ou le dernier appel de processChanges(). Sans cette méthode, les changements ne sont pas calculés jusqu'à la prochaine exécution de reconnaissance. Cette méthode doit être utilisée pour des changements isolés. Lorsqu'il s'agit de plusieurs mises à jour, telles que des opérations de chargement en bloc, utilisez les méthodes startBulkload() et endBulkload().

Tableau 9. Méthodes d'historique des changements (suite)

Méthode	Description
getChangedClasses(long start, long end, int changeType)	Renvoie un tableau de types de classe qui ont été modifiés entre les dates de début et de fin, en fonction d'une valeur de temps pertinente. Le type de changement spécifié peut être l'un des suivants : • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangedClassesForDeltaSynching(long start, long end, int changeType)	Renvoie un tableau de types de classe qui ont été modifiés entre les dates de début et de fin. Le type de changement spécifié peut être l'un des suivants : • DataApi.CREATED • DataApi.DELETED • DataApi.UPDATED • DataApi.UPDATECREATE • DataApi.ANYCHANGE
getChangeHistory(Guid[] Guids, long start, long end, int offset, int nextBatch)	Renvoie l'historique des changements pour la liste spécifiée des identificateurs globaux uniques. Le paramètre nextbatch indique la taille du lot d'enregistrements à renvoyer, en commençant par le décalage spécifié. Cela permet une approche évolutive pour renvoyer des informations de l'historique des modifications.

Gestion de la reconnaissance :

Les méthodes de reconnaissance vous permettent de gérer les exécutions de reconnaissance. Les méthodes de reconnaissance permettent de démarrer et d'arrêter les opérations de reconnaissance, et d'activer ou de désactiver les événements de mise à jour. Elles permettent également d'obtenir l'état de la reconnaissance, et de supprimer l'ensemble des éléments de reconnaissance de la topologie.

Le tableau 10 décrit les méthodes de reconnaissance que vous pouvez utiliser.

La plupart des méthodes suivantes se rapportent à la reconnaissance à équilibrage de charge. Pour plus d'informations, voir «Commande de reconnaissance d'équilibrage de charge», à la page 123.

Tableau 10. Méthodes de reconnaissance

Méthode	Description
abortDiscovery()	Interrompt l'opération de reconnaissance en cours.
getStatus()	Renvoie le statut de l'opération de reconnaissance en cours. Le statut accepte les valeurs suivantes : • `En cours d'exécution` • `En sommeil`

Tableau 10. Méthodes de reconnaissance (suite)

Méthode	Description
startDiscovery(String[] scope, String runName, String locationTag)	Démarre une nouvelle reconnaissance avec une portée. La portée peut être un nom ou contenir des intervalles, des réseaux et des adresses IP, avec des adresses IP spécifiques incluses et exclues.
startDiscovery(RunDefinition runDef, String runName)	Démarre une nouvelle reconnaissance basée sur une définition de l'exécution de la reconnaissance qui indique le profil, la portée, le nom de l'exécution et les informations de balise d'emplacement.
startDiscovery(Guid[] guidList, String runName)	Démarre une nouvelle reconnaissance des objets à l'aide des identificateurs globaux uniques spécifiés.
abortDiscovery(long runId)	Interrompt l'exécution de la reconnaissance spécifiée.
Fix Pack 2 LoadbalancedDiscoveryStatus loadbalancedDiscoveryStatus()	Renvoie le statut de l'opération de reconnaissance à équilibrage de charge en cours. Le statut accepte les valeurs suivantes : • `En cours d'exécution` • `En sommeil`
Fix Pack 2 startLoadbalancedDiscovery(String scopeGroupName, String discoveryPoolName, String profileName, String locationTag)	Démarre une nouvelle reconnaissance avec équilibrage de charge sur chaque portée incluse dans l'attribut `scopeGroup` et sur chaque serveur appartenant au serveur de reconnaissance spécifié dans l'attribut `poolName`.
Fix Pack 2 startLoadbalancedDiscovery(String[] fileNames, int maxScopeSize, String profileName, String locationTag)	Démarre une nouvelle reconnaissance à équilibrage de charge sur chaque portée en commençant par un fichier spécifique. Chaque fichier est analysé, et un nouveau groupe nommé d'après le nom du fichier (sans extension) est créé. Dans chaque groupe, des portées sont ajoutées avec la taille maximale spécifiée dans l'attribut `maxScopeSize`. L'attribut `poolName` est supposé être identique à l'attribut `groupName`.
Fix Pack 2 stopLoadbalancedDiscovery(String discoveryPoolName)	Arrête la reconnaissance à équilibrage de charge pour l'attribut `poolName` spécifié.
Fix Pack 2 pauseLoadbalancedDiscovery(String discoveryPoolName)	Met en pause la reconnaissance à équilibrage de charge pour l'attribut `poolName` spécifié. Toutes les portées de l'exécution de la reconnaissance reviennent à l'état "forTake". Les reconnaissances en cours sont abandonnées.
Fix Pack 2 resumeLoadbalancedDiscovery(String discoveryPoolName)	Renvoie l'objet LoadBalancedDiscoveryStatus qui expose tous les pools de reconnaissance qui sont en cours, y compris ceux en pause. En outre, cette méthode renvoie des informations sur les portées en cours, et qui attendent dans la file d'attente.

Tableau 10. Méthodes de reconnaissance (suite)

Méthode	Description
Fix Pack 3 startDiscoveryRetID(String[] scope, String runName, String locationTag, String addressSpace)	Démarre une nouvelle reconnaissance avec une portée et renvoie l'ID d'exécution de la reconnaissance. La portée peut être un nom ou peut contenir des intervalles, des réseaux et des adresses IP, avec des adresses IP spécifiques incluses et exclues.

Opérations de recherche :

Les méthodes de recherche vous permettent d'accéder aux objets du Common Data Model. Les méthodes de recherche vous permettent de renvoyer des objets de modèle qui correspondent à une requête spécifique ou de renvoyer des informations sur des éléments gérés spécifiques. Vous pouvez également utiliser les méthodes pour renvoyer les objets qui ont changé au cours d'une période de temps spécifiée.

Le tableau 11 décrit les opérations de recherche que vous pouvez exécuter.

Remarque : La plupart des méthodes de recherche qui utilisent un paramètre de profondeur sont désormais dépréciées car elles ne sont pas adaptées lorsque des requêtes sont lancées sur de grandes quantités de données. Si vous devez rechercher des données dont la profondeur est supérieure à un, utilisez une méthode executeQuery. Chaque méthode executeQuery renvoie un objet DataResultSet à partir duquel vous pouvez extraire des informations sur des objets de modèle et vous pouvez utiliser un curseur pour parcourir les données.

Tableau 11. Méthodes de recherche

Méthode	Description
find(Guid guid, int depth, Guid mss, String[] permissions)	Renvoie des informations sur un élément géré spécifique.
find(Guid Guid, int depth, String[] permissions)	Identique à find(String, int, Guid), sauf que la recherche s'effectue sur l'identificateur global unique d'une instance d'objet spécifique plutôt que sur un ensemble complet d'objets à l'aide d'une requête.
find(String query, int depth, Guid mss, String[] permissions) **Remarque :** Cette méthode est obsolète. Utilisez executeQuery ou une méthode rechercher avec un paramètre fillFlag.	Renvoie des objets de modèle correspondant à la requête spécifiée. Le paramètre de profondeur présente les caractéristiques suivantes : • Une profondeur de 0 renvoie un objet avec uniquement son jeu d'identificateurs globaux uniques. • Une profondeur de 1 renvoie tous les attributs de niveau supérieur ainsi que les objets contenus ayant uniquement leur jeu d'attributs d'identificateur global unique. • Une profondeur définie sur DEPTH_INFINITE localise de façon récurrente tous les attributs jusqu'à ce qu'aucun autre objet ne soit trouvé. Les cycles sont évités.

Tableau 11. Méthodes de recherche (suite)

Méthode	Description
find(String query, boolean fillFlag, Guid mss, String[] permissions)	Renvoie des objets de modèle correspondant à la requête spécifiée. • query - Recherche du langage du modèle pour la sélection et le filtrage des résultats. • fillFlag – Indique s'il convient de renseigner tous les attributs. • mss - Identificateur du système logiciel géré, ou valeur null si manquant • permissions - Liste facultative d'autorisations pour limiter les résultats
findChanges(String query, int depth, long start, long end, int changeType) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeChangesQuery	Renvoie les objets qui ont changé au cours de la période spécifiée pour un type de changement donné.
findChanges(String query, boolean fillFlag, long start, long end, int changeType) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeChangesQuery	Renvoie les objets qui ont changé au cours de la période spécifiée pour un type de changement donné.
findChanges(String query, int depth, long start, long end, int changeType **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeChangesQuery	Renvoie les objets qui ont changé dans la période spécifiée pour un type de changement donné.
findChanges(String query, boolean fillFlag, long start, long end, int changeType) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeChangesQuery	Renvoie les objets qui ont changé dans la période spécifiée pour un type de changement donné.
findCollections(Guid guid, String[] permissions)	Voir le chapitre Gestion des collections.
findImpactedBusinessApplications(Guid[] objects)	Voir le chapitre Présentation.
findImpactedBusinessServices(Guid[] objects)	Voir le chapitre Présentation.
findJDO(String root, String jdoQuery, String jdoVarDecl, int depth, Guid mss, String[] permissions) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeJDOQuery.	Identique à find(String, int, Guid) sauf que la requête doit contenir une requête de données Java Data Object query (JDOQL), avec les déclarations de variables facultatives.
findJDO(String root, String jdoQuery, String jdoVarDecl, boolean fillFlag, Guid mss, String[] permissions) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeJDOQuery.	Identique à find(String, int, Guid) sauf que la requête doit contenir une requête de données Java Data Object query (JDOQL), avec les déclarations de variables facultatives.
findRelationships(Guid managedElementGuid, int direction, String type, int scope, String[] permissions)	Voir la section Gestion des relations.
findRelationships(Guid[] sourceGuids, Guid[] targetGuids, String[] types, int comparisonFlags)	Voir la section Gestion des relations.

Tableau 11. Méthodes de recherche (suite)

Méthode	Description
findXML(String query, int depth, int indent, Guid mss, String[] permissions) **Remarque :** Cette méthode est obsolète. Utilisez la méthode findXML(String,boolean,int,Guid,String[].	Renvoie un document XML contenant une liste de tous les objets correspondant à la requête spécifiée. Cette méthode possède des fonctions identiques à find(String, int, Guid) sauf que les objets sont convertis au format XML.
findXML(Guid Guid, int indent, int depth, String[] permissions)	Identique à findXML(String, int, int, Guid) sauf que la recherche s'effectue sur l'ID de l'instance d'objet spécifique, plutôt que sur un ensemble complet d'objets à l'aide d'une requête.
findXML(String query, boolean fillFlag, int indent, Guid mss, String[] permissions)	Renvoie un document XML contenant une liste de tous les objets qui correspondent à la requête spécifiée.
findCount(String query, String[] permissions)	Renvoie le comptage des objets correspondant à la chaîne de requête MQL spécifiée.
executeQuery(String query, Guid mss, String[] permissions)	Exécute une requête MQL et renvoie une interface du curseur de défilement.
executeQuery(String query, int defaultDepth, Guid mss, String[] permissions)	Exécute une requête MQL et renvoie une interface du curseur de défilement.
executeChangesQuery(String query, int defaultDepth, long start, long end, int changeType, boolean deltaSynching)	Renvoie les objets qui ont changé au cours de la période spécifiée pour un type de changement donné Sinon, possède des fonctionnalités identiques à executeQuery (String,int,Guid,String[]). Si le paramètre **changeType** est défini sur DELETED, alors la clause WHERE de la requête est ignorée. Tous les objets de la classe d'objet de modèle spécifiée qui ont été supprimés dans la période donnée seront renvoyés. Les objets de modèle renvoyés sont les objets de modèle d'interpréteur de commande. Ils ne contiennent aucun attribut. Seul l'identificateur global unique est défini. Si le paramètre **changeType** est défini sur ANYCHANGE, alors en plus de l'exécution de la requête find normale, les objets d'interpréteur de commandes seront renvoyés. Toutefois, pour les objets supprimés, la clause where est ignorée.
findChangesForDeltaSynching(String query, boolean fillFlag, long start, long end, int changeType) **Remarque :** Cette méthode est obsolète. Utilisez la méthode executeChangesQuery.	Renvoie les objets qui ont été conservés dans le magasin de données au cours de la période de temps indiquée pour le type de modification spécifiée.

Systèmes logiciels de gestion :

Les méthodes de systèmes logiciels de gestion vous permettent de gérer les systèmes logiciels de gestion dans un Common Data Model. Les méthodes de systèmes logiciels de gestion permettent d'enregistrer et de supprimer un système

logiciel de gestion dans un Common Data Model. Vous pouvez également extraire des informations sur les systèmes logiciels de gestion qui ont été enregistrés avec la base de données TADDM.

Le tableau 12 décrit les méthodes de système logiciel de gestion que vous pouvez utiliser.

Tableau 12. Méthodes de systèmes logiciels de gestion

Méthode	**Description**
deleteManagementSoftwareSystem(Guid guid)	Supprime un système logiciel de gestion de la base de données TADDM. Les objets et les relations appartenant ou reconnus uniquement par ce système logiciel de gestion sont également supprimés. Toutefois, les objets et les relations appartenant ou reconnus par ce système logiciel de gestion ou un autre ne sont pas supprimés. Seule l'association entre ce système logiciel de gestion et les objets et relations lui appartenant est supprimée.
getManagementSoftwareSystemLinks(Guid guid, Guid mss, String[] permissions)	Renvoie les jetons source d'un élément géré ou d'une relation. Si aucun système logiciel de gestion n'est spécifié, la méthode renvoie les jetons source pour chaque système logiciel de gestion qui reconnaît ou possède l'objet spécifié.
getManagementSoftwareSystems(Guid guid, String[] permissions)	Reçoit un tableau de systèmes logiciels de gestion qui ont été enregistrés avec la base de données TADDM. Vous avez la possibilité de fournir l'identificateur global unique d'un élément géré ou d'une relation pour renvoyer uniquement les systèmes logiciels de gestion qui possèdent l'élément géré ou la relation.
registerManagementSoftwareSystem (ManagementSoftwareSystem mss)	Enregistre un nouveau système logiciel de gestion avec la base de données TADDM. Prenez connaissance des détails suivants : • Cette méthode insère uniquement un objet. Elle ne remplace ni ne met à jour un objet existant. • Le modèle doit contenir les clés pour l'objet. • Quand une clé fait référence à un objet parent, ce dernier doit exister. • Quand l'objet fait directement référence à l'identificateur global unique, le développeur de logiciel a la responsabilité de garantir l'existence de cet identificateur global unique dans la base de données TADDM.

Tableau 12. Méthodes de systèmes logiciels de gestion (suite)

Méthode	Description
updateManagementSoftwareSystem (ManagementSoftwareSystem mss)	Met à jour ou insère un système logiciel de gestion dans la base de données. Prenez connaissance des détails suivants : • Cette méthode permet d'insérer, de remplacer ou de mettre à jour un objet existant. • Le modèle doit contenir les clés pour l'objet. • Quand une clé fait référence à un objet parent, ce dernier doit exister. • Quand l'objet fait directement référence à l'identificateur global unique, le développeur de logiciel a la responsabilité de garantir l'existence de cet identificateur global unique dans la base de données TADDM.

MSSObjectLink :

MSSObjectLink est une association entre un système logiciel de gestion et un élément géré qu'il possède.

Les méthodes MSSObjectLink peuvent servir à récupérer tous les MSSObjectLinks correspondant au système logiciel de gestion donné et aux éléments gérés. Ces MSSObjectLinks représentent tous les éléments gérés possédés par un système logiciel de gestion donné ou tous les systèmes logiciels de gestion possédant un élément géré donné. MSSObjectLink n'est pas stocké comme objet de modèle et vous ne pouvez donc pas utiliser l'`interface de programme d'application de recherche` pour obtenir les objets MSSObjectLink. A la place, utilisez les interfaces de programme d'application décrites dans le tableau 13, avec l'interface de programme d'application getManagementSoftwareSystemLinks .

Tableau 13. MSSObjectLink

Méthode	Description
getObjectSourceSystems(Guid[] objectId, String[] permission)	Permet de renvoyer tous les systèmes logiciels de gestion possédant les relations ou les éléments gérés. Cette méthode renvoie un groupe d'objets MSSObjectLink correspondant au système logiciel de gestion qui possède un élément géré. Vous pouvez transmettre un maximum de 50 éléments gérés simultanément dans le groupe d'identificateurs globaux uniques.

Tableau 13. MSSObjectLink (suite)

Méthode	Description
getObjectSourceSystems(Guid objectId, Guid mssGuid, String[] permission)	Permet de renvoyer un groupe d'objets MSSObjectLink, dont chacun correspond au jeton source d'un objet possédé par le système logiciel de gestion donné. Si seul le mssGuid est spécifié, cette méthode renvoie les jetons de source de tous les objets qu'il possède. Par ailleurs, si vous spécifiez uniquement l'identificateur global unique de l'objet, la méthode renvoie les jetons source de tous les systèmes logiciels de gestion qui le possèdent.
getObjectSourceSystems(Guid mssGuid, String mssSourceToken)	Permet de renvoyer un objet MSSObjectLink qui correspond à un élément géré possédé par un système logiciel de gestion donné et identifié par son jeton source (le jeton source doit être spécifié).
getObjectSourceSystems(String joinQuery)	Permet de renvoyer un groupe d'objets MSSObjectLink répondant à la requête JOIN entre un MSSObjectLink et tout autre objet de modèle. Comme les objets MSSObjectLinks seront stockés directement en tant que données relationnelles et non comme objets de modèle, vous ne pourrez plus utiliser le langage MQL pour joindre (JOIN) les objets MSSObjectLink à d'autres objets de modèle. La requête joinQuery transmise dans cette méthode deviendra plutôt une requête SQL. Les attributs spécifiés dans la clause SELECT de la requête SQL doivent être les attributs de l'objet MSSObjectLink puisque cette méthode renvoie un groupe d'objets MSSObjectLink. Exemple : Obtenir tous les systèmes informatiques (ComputerSystems) appartenant à un système logiciel de gestion donné. **SELECT mssobjlink_rel.obj_x,** mssobjlink_rel.msssourcetoken_x, mssobjlink_rel.guid_x, **FROM mssobjlink_rel, compsys** **WHERE mssobjlink_rel.obj_x = compsys.guid_x** **AND mssobjlink_rel.mss_x='5D65T789UK3'**

Gestion de listes d'accès :

Les méthodes de listes d'accès vous permettent de créer et gérer les entrées de listes à partir de l'interface de programme d'application Java. Les applications de fournisseur peuvent gérer les identités à l'aide de ces méthodes.

Ces méthodes d'interface de programme d'application permettent d'effectuer les tâches suivantes :

Créer et supprimer une entrée de liste d'accès
Vous pouvez créer et supprimer les entrées de liste d'accès pour les gérer de façon automatique.

Mettre à jour les propriétés d'une entrée de liste d'accès
Vous pouvez mettre à jour les propriétés des entrées de liste d'accès. Vous pouvez utiliser l'interface de programme d'application pour modifier le mot de passe.

Obtenir les propriétés de l'entrée de liste d'accès
Vous pouvez obtenir les propriétés spécifiques des entrées de liste d'accès dans la fenêtre Liste d'accès de la console de gestion de reconnaissance. L'interface de programme d'application ne permet pas de récupérer le mot de passe.

Vérifier la valeur de propriété de l'entrée de liste d'accès
Vous pouvez vérifier si la valeur de la propriété donnée correspond à celle de l'entrée existant dans la liste d'accès. Vous pouvez utiliser l'interface de programme d'application pour vérifier le mot de passe de l'entrée de la liste d'accès.

Le tableau 14 décrit les méthodes de liste d'accès que vous pouvez utiliser.

Remarque : Les entrées de liste d'accès enregistrées dans le profil de reconnaissance ne sont pas prises en charge par l'interface de programme d'application fourni.

Tableau 14. Méthodes de liste d'accès

Méthode	Description
deleteDiscoveryAccessEntry(String authClassName, String name)	Permet de supprimer l'entrée d'accès de la classe et du nom spécifiés.
getAllDiscoveryAccessEntries()	Permet d'obtenir toutes les entrées d'accès de la reconnaissance.
getDiscoveryAccessEntry (String authClassName, String name)	Permet d'obtenir l'entrée d'accès de la classe et du nom spécifiés.
updateDiscoveryAccessEntry(DiscoveryAccessEntry discoveryAccess)	Permet de mettre à jour les propriétés de l'entrée d'accès de la classe et du nom spécifiés.
verifyDiscoveryAccessEntry(DiscoveryAccessEntry discoveryAccess)	Permet de vérifier si les propriétés données correspondent à celles de l'entrée d'accès de la classe et du nom spécifiés.

Exemple de code Java

1. L'exemple de code Java suivant montre comment créer la connexion, la session et l'instance d'interface de programme d'application :

```
CMDBApi api;
try {
    ApiConnection conn_ = ApiFactory.getInstance().getApiConnection("localhost",
   -1,null,false);
    ApiSession session_ = ApiFactory.getInstance().getSession(conn_, user,
      password, CMDB_DEFAULT);
```

```
    api = session_.createCMDBApi();
  } catch (ApiException ex) {
    ex.printStackTrace();
  } catch (Exception ex) {
    ex.printStackTrace();
  }
```

2. L'exemple de code Java suivant montre comment créer à l'aide d'un programme des entrées d'accès de reconnaissance :

```
try {
    // WebSphere access entry
    DiscoveryAccessEntry entry = new DiscoveryAccessEntry
     (DiscoveryAccessEntry.AUTHCLASS_WEBSPHERE, "user3-websphere");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope3");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 3);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "wasadmin");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.
      PROPERTY_TRUSTSTOREFILECONTENTS, new byte[] { 0x10, 0x20, 0x30 });
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_TRUSTSTOREPASSPHRASE,
   "password");
    entry = api.updateDiscoveryAccessEntry(entry);

    // SNMP access entry
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_SNMP,
   "user4-snmp");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope4");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 4);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_COMMUNITYSTRING, "public");
    entry = api.updateDiscoveryAccessEntry(entry);

    // SNMPv3 access entry
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_SNMPV3,
   "user5-snmpv3");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope5");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 5);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "snmp");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_AUTHPROTOCOL, "MD5");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PRIVPASSWORD,
   "privpassword");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Cisco access entry
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_CISCO,
   "user6-cisco");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope6");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 6);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "cisco");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty("enablepassword", "enablepassword");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ENABLEPASSWORD,
   "enablepassword");
    entry = api.updateDiscoveryAccessEntry(entry);

    // ccmsserver access entry
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_
      CCMSSERVER, "user7-sapccms");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope7");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 7);
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "ccms");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
    entry.setProperty(DiscoveryAccessEntry.PROPERTY_CLIENTID, "clientid");
    entry = api.updateDiscoveryAccessEntry(entry);

    // Computer system access entry
    entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST,
```

```
  "user1-host");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope1");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 1);
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "root");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
   entry = api.updateDiscoveryAccessEntry(entry);

   // Windows computer system access entry
   entry = new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_
     WINDOWSHOST, "user2-winhost");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope2");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "Administrator");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");
   entry.setProperty(DiscoveryAccessEntry.PROPERTY_AUTHTYPE,
     "authType_default");
   entry = api.updateDiscoveryAccessEntry(entry);
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

3. L'exemple de code Java suivant montre comment obtenir toutes les entrées d'accès de reconnaissance :

```
try {
    DiscoveryAccessEntry entry;
    DiscoveryAccessEntry[] entries = api.getAllDiscoveryAccessEntries();
    for (int i = 0; i < entries.length; i++) {
    entry = entries[i];
    String authClassName = (String) entry.getAuthClassName();
    Integer order = (Integer) entry.getProperty(DiscoveryAccessEntry.
    PROPERTY_ORDER);
    switch (order.intValue()) {
    case 1:
      // DiscoveryAccessEntry.AUTHCLASS_HOST.equals(authClassName));
      break;
    case 2:
      // DiscoveryAccessEntry.AUTHCLASS_WINDOWSHOST.equals(authClassName));
      break;
    case 3:
      // DiscoveryAccessEntry.AUTHCLASS_WEBSPHERE.equals(authClassName));
      break;
    case 4:
      // DiscoveryAccessEntry.AUTHCLASS_SNMP.equals(authClassName));
      break;
    case 5:
      // DiscoveryAccessEntry.AUTHCLASS_SNMPV3.equals(authClassName));
      break;
    case 6:
      // DiscoveryAccessEntry.AUTHCLASS_CISCO.equals(authClassName));
      break;
    case 7:
      // DiscoveryAccessEntry.AUTHCLASS_CCMSSERVER.equals(authClassName));
      break;
    default:
      break;
     }
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

4. L'exemple de code Java suivant montre comment obtenir une entrée d'accès de reconnaissance spécifique :

```
try {
    DiscoveryAccessEntry entry = api.getDiscoveryAccessEntry
      (DiscoveryAccessEntry.AUTHCLASS_SNMP, "user4-snmp");
    String authClassName = (String) entry.getAuthClassName();
    String name = (String) entry.getName();
 } catch (ApiException ae) {
    System.err.println("api exception:" + ae);
 } catch (Exception ex) {
    System.err.println("exception:" + ex);
 }
```

5. L'exemple de code Java suivant montre comment mettre à jour une entrée d'accès de reconnaissance spécifique :

```
try {
  // Create an entry with the same name as an existing entry
  DiscoveryAccessEntry entry = new DiscoveryAccessEntry
   (DiscoveryAccessEntry.AUTHCLASS_HOST, "user1-host");

  // Change the scope
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_SCOPENAME, "scope1c");

  // Change the order
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);

  // Change the username
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_USERNAME, "rootc");

  // Change the password
  entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "passwordc");

  // Update the entry
  entry = api.updateDiscoveryAccessEntry(entry);
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

6. L'exemple de code Java suivant montre comment vérifier une entrée d'accès de reconnaissance :

```
try {
// Create an entry with the same name as the existing entry to verify
DiscoveryAccessEntry entry =
 new DiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST, "user1-host");

// Set the order property to the wrong number
entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 1);

// This result should be false
boolean result = api.verifyDiscoveryAccessEntry(entry));

// Set the order property to the correct number
entry.setProperty(DiscoveryAccessEntry.PROPERTY_ORDER, 2);

// This result should be true
result = api.verifyDiscoveryAccessEntry(entry));

// Set the password property to the wrong value
entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "password");

// This result should be false
result = api.verifyDiscoveryAccessEntry(entry));

// Set the password property to the correct value
entry.setProperty(DiscoveryAccessEntry.PROPERTY_PASSWORD, "passwordc");

// This result should be true
```

```
   result = api.verifyDiscoveryAccessEntry(entry));
} catch (ApiException ae) {
   System.err.println("api exception:" + ae);
} catch (Exception ex) {
   System.err.println("exception:" + ex);
}
```

7. L'exemple de code Java suivant montre comment supprimer une entrée d'accès de reconnaissance :

```
try {
    api.deleteDiscoveryAccessEntry(DiscoveryAccessEntry.AUTHCLASS_HOST,
   "user1-host");
} catch (ApiException ae) {
    System.err.println("api exception:" + ae);
} catch (Exception ex) {
    System.err.println("exception:" + ex);
}
```

Gestion des collections :

Les méthodes de collection vous permettent de gérer les collections dans le Common Data Model. Les méthodes de collection permettent d'ajouter ou de supprimer des membres des collections, et d'extraire toutes les collections auxquelles appartient un élément géré spécifique. Il est important de noter que ces méthodes sont obsolètes. Pour les méthodes équivalentes relatives aux collections personnalisées (CustomCollections), voir la description de chaque méthode obsolète.

Pour plus d'informations, voir «Gestion des modèles de groupement», à la page 86.

Le tableau des méthodes de collection décrit les méthodes de collection que vous pouvez utiliser.

Tableau 15. Méthodes de collection

Méthode	Description
addCollectionMembers(Guid collectionGuid, Guid[] guids)	Ajoute des membres à une collection. Cette méthode n'est pas disponible dans l'interface EJB (Enterprise JavaBeans). Cette méthode est obsolète. Utilisez à la place la méthode updateGroupingPattern(GroupingPattern groupingPattern) suivie de l'une des méthodes runPatternNow(...) .
findCollections(Guid guid, String[] permissions)	Extrait toutes les collections auxquelles appartient l'élément géré spécifique. Une collection peut contenir d'autres collections, mais cette méthode renvoie uniquement la collection de premier niveau à laquelle appartient l'élément géré spécifié. Cette méthode est obsolète. Utilisez à la place la méthode findCustomCollections(Guid guid, String[] permissions).

Tableau 15. Méthodes de collection (suite)

Méthode	Description
removeCollectionMembers(Guid collectionGuid, Guid[] guids)	Supprime des membres d'une collection. Les membres d'une collection peuvent être supprimés en indiquant une grappe d'attributs d'identificateur global unique de collection ou une grappe d'attributs d'identificateur global unique d'élément géré. Quand un membre à supprimer n'est pas actuellement un membre de la collection, il est ignoré. Cette méthode est obsolète. Utilisez à la place la méthode updateGroupingPattern(GroupingPattern groupingPattern) suivie de l'une des méthodes runPatternNow(...) .

Gestion du modèle :

Les méthodes de collection vous permettent de gérer les objets dans le Common Data Model. Les méthodes de modèle permettent d'ajouter, de supprimer et de mettre à jour les objets dans le Common Data Model. Elles permettent également de comparer les objets et de reconstruire les données dérivées, telles que les dépendances, les relations et la consolidation des données.

Le tableau 16 décrit les méthodes de modèle que vous pouvez utilisez.

Tableau 16. Méthodes de gestion de modèle

Méthode	Description
add(ModelObject[] obj, Guid mss)	Ajoute un nouvel objet à la base de données. **Remarque :** Cette méthode ne peut pas être utilisée pour ajouter des objets de modèle GroupingPattern ou Selector. Elle émet une exception API (ApiException) lorsque l'un des objets fournis est du type GroupingPattern ou Selector. Utilisez une interface de programme d'application GroupingPattern pour toutes les opérations sur des GroupingPatterns ou des sélecteurs.
addArrayElements(Guid object, String attrName, Guid[] elements, Guid mss)	Ajoute les éléments à la grappe indiquée de l'objet spécifié sans extraire l'objet ou la grappe.
compare(ModelObject left, ModelObject[] right, CompareOptions opts) compare(ObjectRef obj1, ObjectRef[] objs, CompareOptions opts)	Compare un objet de modèle (ou un document maître golden) à un ensemble d'objets.

Tableau 16. Méthodes de gestion de modèle (suite)

Méthode	Description
delete(Guid[] guids, Guid mss) delete(ModelObject[] obj, Guid mss)	Supprime les objets spécifiés par l'identificateur global unique de la base de données et supprime en cascade tous les objets contenus dans les objets dans l'un des cas suivants : • Aucun système logiciel de gestion n'est fourni. • L'objet est la propriété exclusive du système logiciel de gestion spécifié. L'objet n'est pas supprimé quand un système logiciel de gestion est fourni et que l'objet appartient à un autre système logiciel de gestion. C'est l'association entre l'objet et le système logiciel de gestion qui est supprimée. Quand un objet est supprimé de la base de données TADDM, toutes les relations et les appartenances à la collection associées sont également supprimées. Si l'objet spécifié est un objet de modèle de niveau supérieur, tous les objets contenus sont également supprimés. Par exemple, quand un système informatique est supprimé, le système d'exploitation et les interfaces IP contenus dans l'objet, ainsi que les relations entre le système informatique et les interfaces IP, sont également supprimés. **Remarque :** Cette méthode ne peut pas être utilisée pour supprimer des objets de modèle GroupingPattern ou Selector. Elle émet une exception API (ApiException) lorsque l'un des objets fournis est du type GroupingPattern ou Selector. Utilisez une interface de programme d'application GroupingPattern pour toutes les opérations sur des GroupingPatterns ou des sélecteurs.

Tableau 16. Méthodes de gestion de modèle (suite)

Méthode	Description
deleteStale(Guid mss, long date) **Remarque :** Cette méthode est obsolète.	Supprime les éléments gérés et les relations qui n'ont pas été touchées depuis une date spécifiée. Les éléments gérés et les relations dont l'horodatage de mise à jour est inférieur ou égal à la date spécifiée sont supprimés. Si une relation ou un élément géré périmé appartient uniquement au système logiciel de gestion, l'élément géré ou la relation n'est pas supprimé de la base de données. Seule l'association entre la relation ou l'élément géré avec les systèmes logiciels de gestion spécifié est supprimée. Toutefois, si une relation ou un élément géré périmé appartient uniquement au système logiciel de gestion spécifié, la relation ou l'élément géré est supprimé de la base de données. Toutes les relations et les appartenances à la collection associées à un élément géré supprimé sont également supprimées. Si l'objet spécifié est un objet de modèle de niveau supérieur, tous les objets contenus sont également supprimés. Par exemple, quand un système informatique est supprimé, le système d'exploitation et les interfaces IP contenus dans l'objet, ainsi que les relations entre le système informatique et les interfaces IP, sont également supprimés.
Fix Pack 2 refresh(Guid mss, long date)	Supprime les éléments gérés et les relations qui n'ont pas été stockés depuis une date spécifiée. Les éléments gérés et les relations dont l'horodatage de mise à jour est inférieur à la date spécifiée sont supprimés. Si une relation ou un élément géré périmé appartient à plusieurs systèmes logiciels de gestion, l'élément géré ou la relation est supprimé de la base de données. Toutes les relations et les appartenances à la collection associées à un élément géré supprimé sont également supprimées. Si l'objet spécifié est un objet de modèle de niveau supérieur, tous les objets contenus sont également supprimés. Par exemple, quand un système informatique est supprimé, le système d'exploitation et les interfaces IP contenus dans l'objet, ainsi que les relations entre le système informatique et les interfaces IP, sont également supprimés.
endBulkload(long bulkloadId)	Marque la fin d'une opération de chargement en bloc. Chaque demandeur qui appelle la méthode startBulkload() doit appeler endBulkload() pour déverrouiller le sous-système de stockage.

Tableau 16. Méthodes de gestion de modèle (suite)

Méthode	Description
exportData(File directoryToWriteTo, long maxFileSize, Guid mss)	Exporte tous les objets de la base de données TADDM dans le répertoire spécifié, créant des fichiers pour chaque classe d'objet à l'aide de la méthode find() avec une profondeur illimitée. Quand la taille de fichier maxi en octets est dépassée, un nouveau fichier est créé avec une extension `.N`, N étant incrémenté au besoin. Le format de la sortie respecte le format du schéma XML.
importData(URI source, boolean rebuildTopo, Guid mss)	Convertit les données XML de la source indiquée dans les objets de modèle qui sont mis à jour dans la base de données TADDM, conformément aux règles suivantes : • Quand la source est un fichier, le contenu du seul fichier est lu et inséré. • Quand la source est un répertoire, chaque fichier du répertoire est importé. • Quand la source est un objet éloigné, tel qu'une adresse HTTP, chaque mise à jour s'effectue dans sa propre transaction. Les erreurs annulent la mise à jour de l'objet en cours uniquement et l'importation se poursuit.
rebuildTopology()	Reconstruit les données dérivées de la base de données TADDM, telles que les dépendances, les relations et la consolidation de données. Au cours de cette opération, la totalité de la base de données est verrouillée contre les mises à jour.
removeArrayElements(Guid object, String attrName, Guid[] elements, Guid mss)	Supprime les éléments spécifiés de l'objet donné de la grappe indiquée dans la base de données TADDM sans extraire l'objet ou la grappe du client.
startBulkload(long timeoutInSeconds)	Verrouille le sous-système de stockage contre les autres changements de la base de données, y compris les reconnaissances et les synchronisations. Vous devez appeler cette méthode avant d'effectuer des changements importants. Le verrouillage est conservé jusqu'à ce que la méthode endBulkload() soit appelée.

Tableau 16. Méthodes de gestion de modèle (suite)

Méthode	Description
update(ModelObject obj, mss) update(ModelObject[] obj, mss)	Met à jour ou insère un objet de modèle dans la base de données. Les attributs qui sont définis dans l'objet source sont fusionnés dans la destination, tandis que ceux qui ne sont pas définis ne sont pas mis à jour. Quand un objet de la clé et du type spécifiés n'existe pas, un nouvel objet est créé dans la base de données. Prenez connaissance des détails suivants : • Le nouvel objet doit avoir une clé ou son identificateur global unique défini. Quand une clé fait référence à un objet parent, ce dernier doit exister. Un objet vide qui ne possède que l'identificateur global unique défini est suffisant pour identifier un parent. • Quand l'objet fait directement référence à l'identificateur global unique, le développeur de logiciel a la responsabilité de garantir l'existence de cet identificateur global unique dans la base de données TADDM. • Quand l'identificateur global unique du système logiciel de gestion est une valeur null, les objets sont insérés ou mis à jour dans la base de données TADDM et aucune liaison Système logiciel de gestion-Objet n'est mise à jour. Quand l'identificateur global unique du système logiciel de gestion n'est pas une valeur NULL, la liaison Système logiciel de gestion-Objet est mise à jour. • Vous risquez d'avoir besoin de reconstruire la topologie pour déduire automatiquement les dépendances et les relations explicites du nouvel objet. • Les grappes sont remplacées dans leur intégralité. **Remarque :** Cette méthode ne peut pas être utilisée pour mettre à jour des objets de modèle GroupingPattern ou Selector. Elle émet une exception API (ApiException) lorsque l'un des objets fournis est du type GroupingPattern ou Selector. Utilisez une interface de programme d'application GroupingPattern pour toutes les opérations sur des GroupingPatterns ou des sélecteurs.

Tableau 16. Méthodes de gestion de modèle (suite)

Méthode	Description
updateXML(String xml, Guid mss)	Identique à la méthode update(ModelObject[] obj, mss) excepté que les objets sont représentés comme une chaîne XML. **Remarque :** Cette méthode ne peut pas être utilisée pour mettre à jour des objets de modèle GroupingPattern ou Selector. Elle émet une exception API (ApiException) lorsque l'un des objets fournis est du type GroupingPattern ou Selector. Utilisez une interface de programme d'application GroupingPattern pour toutes les opérations sur des GroupingPatterns ou des sélecteurs.

Exemple

Cet exemple illustre la méthode de comparaison des objets.

```
//Rechercher deux objets comparables à comparer en premier.

ModelObject mo[] = api.find(
      "SELECT * FROM SunSPARCUnitaryComputerSystem", 3, null, null);

if (mo != null) {
   if (mo.length > 1) {
      ModelObject mo1 = mo[0];
      ModelObject mo2 = mo[1];

      try {
         System.out.println("Comparing " + mo1.getDisplayName() +
            " to " + mo2.getDisplayName());
      } catch (Exception e) {
         e.printStackTrace();
      }

      // ObjectRef est une structure de données simple qui contient l'identificateur
      // global unique et la version de l'objet à comparer.

      ObjectRef objectRef2 = new ObjectRef(mo2.getGuid(), 0);
      ComparisonResult result = api.compare(new ObjectRef(mo1.getGuid(), 0),
         new ObjectRef[]{objectRef2},
         new CompareOptions(true));
         handleModel((TreeTableModel)result);
   }
}

public void handleModel(TreeTableModel model) {
   CompareResultRow row = model.getRoot();
   handleRow(model, row, "");
}

private void handleRow(
      TreeTableModel model, CompareResultRow row, String attributeName){
   System.out.println("Handling row " + row);

   int nColumns = model.getColumnCount();

   for (int i = 0; i < nColumns; i++) {
      String columnName = model.getColumnName(i);
      Object value = model.getValueAt(row, i);
      System.out.println("Col Name " + columnName + " value " + value);
```

```
      //Première colonne, il s'agit de attributeName
      if (i == 0) {
         if (!"".equals(attributeName)) {
            attributeName = attributeName + ":" + String.valueOf(value);
         } else {
            attributeName = String.valueOf(value);
         }
      }

      //Calculer le nom de colonne et conserver dans la base de données ici.
   }

   List children = row.getChildren();

   if (children != null) {
      Iterator it = children.iterator();
      while (it.hasNext()) {
         CompareResultRow resultRow = (CompareResultRow) it.next();  //recurse
         handleRow(model, resultRow, attributeName);
      }
   }
}
```

Gestion des relations :

Les méthodes de relation vous permettent de gérer les relations entre les objets du Common Data Model. Elles permettent d'ajouter et de supprimer des relations entre les éléments gérés dans le Common Data Model. Vous pouvez ainsi extraire un graphique de relation pour un type de relation donné.

Le tableau 17 décrit les méthodes de relations que vous pouvez utiliser.

Tableau 17. Méthodes de relations

Méthode	Description
addRelationships(Relationship[] relationships, Guid mss)	Ajoute une ou plusieurs relations à la base de données TADDM. Les éléments gérés source et cible de la relation doivent exister avant d'ajouter une relation entre les éléments. Une instance de relation ne peut pas exister seule dans la base de données TADDM. Elle doit être reconnue par, ou appartenir à un ou plusieurs systèmes logiciels de gestion.
deleteRelationships(Guid[] guids, Guid mss) deleteRelationships(String type, Guid source, Guid target, Guid mss)	Supprime une ou plusieurs relations de la base de données TADDM, dans l'un des cas suivants : • Quand aucun système logiciel de gestion n'est spécifié • Quand une relation appartient uniquement au système logiciel de gestion spécifié La relation n'est pas supprimée quand un système logiciel de gestion est indiqué et qu'une relation appartient à un autre système logiciel de gestion. Dans ce cas là, seule l'association entre la relation et le système logiciel de gestion est supprimée.

Tableau 17. Méthodes de relations (suite)

Méthode	Description
findRelationships(Guid managedElementGuid, int direction, String type, int scope, String[] permissions)	Extrait le graphique de relation pour le type de relation donné à partir de l'élément géré spécifié. Les relations stockées dans la base de données TADDM peuvent être traversées dans les directions suivantes : • En partant d'un élément géré source pour aller vers l'avant • En partant d'un élément géré cible pour aller vers l'arrière
findRelationships(Guid[] sourceGuids, Guid[] targetGuids, String[] types, int comparisonFlags) findrelationships(managedElementGuid => findRelationships(managedElementGuid	Extrait les relations contenant uniquement des informations de base. Cette méthode s'exécute plus rapidement que la méthode findrelationships(managedElementGuid, direction, type, scope, permissions).

Gestion des sessions :

Les méthodes de session vous permettent d'établir des connexions et de gérer des sessions avec le serveur. Ces méthodes de session permettent d'ouvrir et de fermer des sessions avec le serveur TADDM et de récupérer la connexion en cours.

Le tableau 18 décrit les méthodes de session que vous pouvez utiliser.

Tableau 18. Méthodes de session

Méthode	Description
close()	Ferme une session.
ApiFactory.getInstance().getApiConnection (String host, int port, String trustStoreLocation, boolean useSSL)	Créez une connexion avec l'hôte et le port spécifiés. Si la valeur du port spécifiée est `-1`, le port par défaut spécifié dans le fichier `collation.properties` est utilisé. Le paramètre trustStoreLocation indique l'emplacement du fichier de certificat à utiliser pour les connexions SSL.
ApiFactory.getInstance().getSession (ApiConnection conn, String user, String password, long version)	Renvoie un objet de session qui est utilisé pour exécuter les méthodes de l'interface de programme d'application TADDM.
ApiFactory.getInstance().getSession (ApiConnection conn, long sessionId, long version)	Renvoie un objet de session qui est utilisé pour exécuter les méthodes de l'interface de programme d'application TADDM.
release()	Cette méthode n'étant pas prise en charge, utilisez la méthode `close()`.

Exemples

La connexion au serveur TADDM implique l'établissement d'une connexion sur le serveur et l'ouverture d'une session avec un compte et un mot de passe utilisateur. L'exemple suivant illustre comment établir une connexion au serveur :

```
private ApiConnection conn_;
   try {
      conn_ = ApiFactory.getInstance().getConnection(
         "host.abcxyz.com",            //host name
```

```
        9433,                        //port number
         null,                        //Location of jssecacerts.cert file for SSL
         false);                      //true for SSL, false for non SSL
   } catch (Throwable th) {
        th.printStackTrace();
}
```

Sinon, vous pouvez établir une connexion SSL, comme l'illustre l'exemple suivant :

```
private ApiConnection conn_;
   try {
      conn_ = ApiFactory.getInstance().getConnection(
         "host.abcxyz.com",            //host name
        9433,                        //port number
         "c:\\temp\\jssecacerts.cert", //Location of jssecacerts.cert file for SSL
         true);                       //true for SSL, false for non SSL
   } catch (Throwable th) {
        th.printStackTrace();
}
```

Le serveur TADDM utilise par défaut le port 9433, mais vous pouvez indiquer un autre port en spécifiant une valeur appropriée dans le fichier `$COLLATION_HOME/etc/collation.properties`.

Lors de l'établissement d'une connexion SSL, vous devez spécifier l'emplacement du fichier `jssecacerts.cert`. Téléchargez ce fichier depuis le portail Web Java du serveur TADDM, disponible à l'adresse suivante : http://*server_name*:*web_server_port*, par exemple http://localhost:9430.

Une fois la connexion ApiConnection établie, ouvrez une session sur le serveur, comme illustré par l'exemple suivant :

```
String user = "smartoperator";     // login user name
String password = "foobar";        // user password
long version = 0;                  // version
ApiSession session_ = ApiFactory.getInstance().getSession(conn_, user, password,
  version);
CMDBApi api = session_.createCMDBApi();
```

CMDBApi désigne la référence éloignée qui permet d'effectuer toutes les opérations liées à l'interface de programme d'application sur le serveur TADDM.

Gestion des transactions :

L'interface de programme d'application de transaction est obsolète.

Les méthodes suivantes génèrent uniquement des messages d'avertissement et aucune transaction ne sera démarrée, validée ou annulée :

beginTransaction()

beginTransaction(int timout)

commitTransaction()

rollbackTransaction()

Gestion des versions :

Les méthodes de version vous permettent de gérer les versions de données de la base de données TADDM. Les demandes de version permettent de créer des

images instantanées convenues des données actuelles de la base de données TADDM, de supprimer des versions et d'établir la liste des versions définies disponibles.

Le tableau 19 décrit les méthodes de version que vous pouvez utiliser.

Tableau 19. Méthodes de version

Méthode	Description
createEmptyVersion(name, description)	Crée une version vide sans aucune donnée.
createVersion(name, description)	Crée une image instantanée convenue des données actuelles de la base de données TADDM.
deleteVersion(versionID)	Supprime la version indiquée de la base de données TADDM.
getAllVersions()	Renvoie les noms de toutes les versions de données de la base de données TADDM définies.
getVersion()	Renvoie l'objet TopologyVersion de la version actuelle des données de la base de données TADDM affichée.

Métadonnées :

Les méthodes de métadonnées permettent de gérer les métadonnées dans le Common Data Model. Ces méthodes permettent d'ajouter, de mettre à jour ou de supprimer les attributs étendus et de définir les valeurs associées. Vous pouvez également les utiliser pour renvoyer les informations de métadonnées dans le Common Data Model y compris le nombre, les types et les noms de tous les attributs d'un objet de modèle.

Le tableau 20 décrit les méthodes de métadonnées que vous pouvez utiliser.

Tableau 20. Méthodes de métadonnées

Méthode	Description
defineExtendedAttributeMeta(UserDataMeta udm)	Ajoute ou met à jour les métadonnées des attributs étendus.
getAllMetaData()	Renvoie les métadonnées relatives au Common Data Model sans son composant Simplified Model. Elle n'affiche aucune classe du Simplified Model ni aucun déplacement ou changement des attributs et relations.
getAllMetaData(boolean flatten, Locale locale, boolean skipSimplifiedModel)	Renvoie toutes les métadonnées. Cette méthode est équivalente à find("ObjectClass", ...) où un cache de métadonnées est utilisé sur le serveur pour un accès plus rapide.
getClassNames()	Renvoie une grappe de noms abrégés de la classe de modèle, paires de noms qualifiés complets.

Tableau 20. Méthodes de métadonnées (suite)

Méthode	Description
getExtendedAttributeMeta(String classname)	Extrait les métadonnées des attributs étendus d'une classe. Cette méthode extrait les métadonnées des attributs étendus de la catégorie `taddm_global` et de la catégorie personnalisée. Les objets `UserDataMeta` sont collectés à partir de la classe spécifiée et à partir de tous les éléments qui, dans la hiérarchie, sont supérieurs à la classe spécifiée.
getExtendedAttributes(Guid objGuid)	Extrait les valeurs d'attribut étendu pour un objet. **Remarque :** Cette méthode est obsolète.
getMetaData(String className)	Renvoie le nombre, les types et les noms de tous les attributs dans l'objet de modèle. Cela comprend la règle de nom/clé, le confinement, la relation et les informations de type énumératif.
removeExtendedAttributeMeta(String classname, Guid acct)	Supprime les attributs étendus sur toute la classe ou les attributs étendus pour un compte et une classe spécifiés. **Remarque :** Cette méthode est obsolète.
setExtendedAttributes(Guid objGuid, AttrNameValue[]	Définit les valeurs des attributs étendus. **Remarque :** Cette méthode est obsolète. Utilisez à la place l'attribut `XA` des objets.

Présentation :

Les méthodes de présentation vous permettent de déterminer les systèmes concernés et de renvoyer des informations de topologie. Les méthodes de présentation permettent de déterminer les applications métier et les services concernés, selon des éléments de configuration spécifiques. Elles permettent également de renvoyer des topologies, des graphiques et des détails d'objet de modèle.

Le tableau 21 décrit les méthodes de présentation que vous pouvez utiliser.

Tableau 21. Méthodes de présentation

Méthode	Description
findImpactedBusinessApplications(Guid[] objects)	Détermine les applications métier qui sont concernées, selon la grappe spécifiée d'éléments de configuration.
findImpactedBusinessServices(Guid[] objects)	Détermine les services métier qui sont concernés, selon la grappe spécifiée d'éléments de configuration.
getDetailsPanel(ObjectRef ref)	Renvoie la liste des détails pour la référence d'objet spécifiée. Une référence d'objet est la combinaison de l'identificateur global unique et de la version d'un objet.

Tableau 21. Méthodes de présentation (suite)

Méthode	Description
getGraphView(ViewDefiner graphView)	Renvoie un graphique pour le paramètre ViewDefiner spécifié qui décrit la vue graphique.
getGraphViewImage(ViewDefiner graphView) **Remarque :** Fix Pack 3 Cette méthode est obsolète.	Renvoie un objet ImageStream pour le paramètre ViewDefiner spécifié qui décrit la vue graphique.
getTreeView(ViewDefiner treeView)	Renvoie un objet TopologyTreeModel pour le paramètre ViewDefiner spécifié qui décrit l'arborescence.

Exemple de code Java

- L'exemple de code Java suivant illustre comment extraire la liste des caractéristiques des objets à l'aide de l'identificateur global unique.

  Remarque :

  – L'identificateur global unique permet d'identifier les objets stockés dans la base de données.

  – ObjectRef est une structure de données simple qui contient l'identificateur global unique et la version de l'objet pour lequel vous souhaitez extraire la liste des caractéristiques.

  – Vous pouvez extraire l'identificateur global unique de l'objet dont vous avez besoin en entrant une commande semblable à celle qui suit. Cette commande particulière permet d'obtenir la liste de tous les objets associés à ComputerSystem.

  ```
  SELECT * FROM ComputerSystem
  ```

```
ModelObject mo[] = api.find(
      "SELECT * FROM ComputerSystem",
      1,
      null,
      null);

if (mo != null) {
   for (int i=0; i<mo.length; i++){
      Guid guid = mo[i].getGuid();
      System.out.println("Getting details panel for " + guid);
      DetailPanelModel model = api.getDetailsPanel(
         new ObjectRef(guid,0));     //guid and version
      System.out.println("model is " + model);
   }
}
```

- L'exemple suivant illustre comment extraire des vues graphiques.

  L'énumération des graphiques et des arborescences est définie dans la classe com.collation.proxy.api.presentation.common.ViewDefinerEnum.

```
ViewDdefiner viewDefiner = new ViewDefiner(
      ViewDefinerEnum.GRAPH_APPLICATION_PHYSICAL_INFRASTRUCTURE,
            VersionedObject.DYNAMIC);
TopologyGraphModel gv = api.getGraphView(viewDefiner);
```

- L'exemple suivant illustre comment rechercher les applications et les services métier concernés :

```
// Rechercher les objets pour l'analyse de l'incidence

ModelObject mo[] = api.find("SELECT * FROM ApacheServer", 1, null, null);

if (mo != null) {
   Application[] applications = api.findImpactedBusinessApplications(
         new Guid[]{mo[0].getGuid()});

   if (applications != null) {
      for (int i=0;i<applications.length;i++) {
         System.out.println(applications[i].getDisplayName());
      }
   }

   BusinessSystem[] systems = api.findImpactedBusinessServices(
         new Guid[]{mo[0].getGuid()});
   if (systems != null) {
      for (int i=0;i<systems.length;i++) {
         System.out.println(systems[i].getDisplayName());
      }
   }
}
```

Sécurité :

Les méthodes de sécurité vous permettent de gérer les autorisations, les droits et les rôles dans la base de données TADDM. Les méthodes de sécurité permettent d'ajouter et de supprimer des autorisations, et de déterminer les autorisations et les droits d'utilisateurs spécifiques. Vous pouvez également utiliser ces méthodes pour définir les rôles affectés à un utilisateur et déterminer si un utilisateur a accès à une ou plusieurs opérations d'exécution.

Remarque : Les méthodes de sécurité fonctionnent sur les objets CustomCollection avec l'attribut hierarchyType défini sur "AccessCollection". Elles ne prennent pas en charge les anciens objets AccessCollection.

Le tableau 22 décrit les méthodes de sécurité que vous pouvez utiliser.

Tableau 22. Méthodes de sécurité

Méthode	Description
addAccess(Principal user, Resource resource, String role, String[] permissions)	Ajoute des autorisations pour un objet propre à un rôle.
addRuntimeAccess(Principal user, String role, String[] permissions)	Ajoute des autorisations pour une ou plusieurs opérations d'exécution à un rôle.
assignPersonInRoleToAccessCollection(Person user, Role role, Guid[] guids, long[] versionId)	Crée une affectation (dans des versions potentiellement multiples) entre une personne d'un rôle et une liste de collections d'accès.
deleteAccess(Principal user, Resource resource, String role, String[] permissions)	Supprime un droit d'accès pour une collection spécifique d'un rôle.
deleteRuntimeAccess(Principal user, String role, String [] permissions)	Supprime des autorisations pour une ou plusieurs opérations d'exécution d'un rôle.
getAccessDecisions(Principal user, Resource[] resources, String[] permissions)	Détermine si le demandeur peut accéder à un ou plusieurs objets avec les autorisations spécifiées.

Tableau 22. Méthodes de sécurité (suite)

Méthode	Description
getDataPermissions(Principal user, Resource[] resources)	Détermine les autorisations au niveau des données accordées à un utilisateur pour un ensemble d'objets.
getEntitlements(Principal user, String[] permissions)	Extrait les droits de l'utilisateur. Les droits sont les objets auxquels l'utilisateur peut accéder, selon les stratégies de sécurité définies.
getRoles(Principal user)	Extrait les rôles affectés à un utilisateur.
getRuntimeAccess(Principal user)	Extrait les autorisations pour les opérations d'exécution d'un utilisateur.
getRuntimeAccessDecisions(Principal user, String[] permissions)	Détermine si un utilisateur possède un accès à une ou plusieurs opérations d'exécution.
removePersonInRoleToAccessCollection(Person user, Role role, Guid[] guids, long[] versionId)	Supprime une affectation dans des versions potentiellement multiples, en se basant sur la personne, le rôle et la liste de collections d'accès spécifiés auxquels la personne qui possède le rôle est affectée.
addAccess(Principal user, AccessDefinition[] accessDefinition)	Ajoute un ou plusieurs objets (chacun avec un ensemble d'autorisations) à un rôle, comme indiqué par chaque objet AccessDefinition.
addDataPermissionToRole(String role, String permission)	Autorise des données pour un rôle à chaque fois qu'un rôle existe dans les règles enregistrées.
addRuntimePermissionToRole(String role, String permission)	Ajoute une autorisation d'exécution à un rôle.
deletePermission(String permission)	Supprime une autorisation, qu'elle existe ou non dans les règles enregistrées.
deletePermissionFromRole(String role, String permission)	Supprime une autorisation pour un rôle à chaque fois qu'un rôle existe dans les règles enregistrées.
deleteRole(String role)	Supprime une autorisation pour un rôle à chaque fois qu'un rôle existe dans les règles enregistrées.
getEntitlementsForRole(Principal user, String role)	Extrait les autorisations pour l'utilisateur dans un rôle spécifié.

Gestion des modèles d'application :

Les méthodes de modèle d'application activent la création, la modification, la suppression et la récupération des modèles d'application et des règles.

Important : Cette interface de programme d'application est obsolète pour la version 7.3 de TADDM et toute version ultérieure. L'interface de programme d'application GroupingPatternAPI peut être utilisée pour gérer les services métier. Cette interface de programme d'application fournit des méthodes de création, modification et suppression de Modèles de Groupement. Pour plus d'informations, voir «Gestion des modèles de groupement», à la page 86.

Les modèles d'application spécifient les règles MQL qui sont périodiquement appliquées à la base de données TADDM pour définir des applications métier ou des collections. Chaque modèle indique une ou plusieurs règles avec les attributs suivants :

MQLRuleName
Le nom de la règle MQL. Cet attribut est obligatoire.

FunctionalGroupName
L'objet groupe fonctionnel contient ce que la requête MQL renvoie. Cet attribut est requis pour toute règle définissant une application métier. Il n'est pas utilisé pour les collections définies par les règles.

MQLQuery
La requête MQL à exécuter. Les objets renvoyés par la requête sont ajoutés à l'application métier ou à la collection.

Le tableau 23 décrit les méthodes du modèle d'application que vous souhaitez utiliser.

Tableau 23. Les méthodes du modèle d'application

Méthode	Description
createAppTemplate(String name, String type, boolean removeNonMembers, MQLRule[] operators)	Créée un modèle avec les règles spécifiées. Les paramètres suivants sont disponibles : **nom** Le nom du modèle d'application à créer. **type** Le type de modèle à créer (application, collection ou service). Les valeurs valides sont «application» pour l'application métier, «Collection» pour les collections ou «Service» pour les services métier. **removeNonMembers** Une valeur booléenne indique si des objets existant dans l'application métier ou la collection peuvent être supprimés lorsqu'ils ne correspondent plus aux règles du modèle. **MQLRule** Un tableau contenant une ou plusieurs règles.

Tableau 23. Les méthodes du modèle d'application (suite)

Méthode	Description
updateAppTemplate(String name, String type, boolean removeNonMembers, MQLRule[] operators)	Met à jour un modèle avec les règles spécifiées. Les paramètres suivants sont disponibles : **nom** Le nom du modèle d'application à mettre à jour. **type** Le type de modèle à mettre à jour (application, collection ou service). Les valeurs valides sont «application» pour l'application métier, «Collection» pour les collections ou «Service» pour les services métier. **removeNonMembers** Une valeur booléenne indique si des objets existant dans l'application métier ou la collection peuvent être supprimés lorsqu'ils ne correspondent plus aux règles du modèle. **MQLRule** Un tableau contenant une ou plusieurs règles.
getAllAppTemplates()	Extrait tous les modèles d'application.
getAppTemplate(String name, int type)	Récupère le modèle d'application pour le nom et le type spécifiés.
removeAppTemplate(String name, int type)	Supprime le modèle d'application pour le nom et le type spécifiés.
removeMQLRule(String name)	Supprime une règle MQL. Une règle peut être supprimée uniquement si elle n'est pas associée à des modèles d'application.

Création d'un modèle d'application métier

Pour créer un modèle d'application pour une application métier, procédez comme suit :

1. Créez une application métier et attribuez-lui le nom transmis via la ligne de commande. Par exemple : `TADDM - Production`. Récupérez l'identificateur global unique renvoyé.
2. Définissez le nom du modèle d'application en respectant le format suivant :
   identificateur_global_unique_application_métier:*nom_app_métier*

   Par exemple,
   `AA0A20EE5BBD336481279CA664FB380A:TADDM - Production`
3. Précédez les noms de règle de l'identificateur global unique de l'application métier comme préfixe, mais veillez à ne pas inclure de signe deux-points dans le nom de règle. Par exemple :
   `AA0A20EE5BBD336481279CA664FB380Adatabase`

Exemple : Création d'un modèle d'application métier

L'exemple suivant crée un modèle d'application métier, les règles MQL et l'application métier associée :

```
# création de l'application métier "TADDM - Production"
BAname = "TADDM - Production"

# obtention de l'identificateur global unique de l'application métier
myBA = ModelFactory.newInstance(Class.forName
("com.collation.platform.model.topology.app.Application"))
myBA.setName(BAname)
appGuid = api.update(myBA,None)
MQLRuleClass = Class.forName("com.collation.platform.model.apptemplate.MQLRule")
rules = [ModelObjectFactory.newInstance(MQLRuleClass)]

# création d'un nom de règle MQL requis par TADDM, tel que AA0A20EE5BBD336481279CA664FB380Adatabase
rulename = "database"
RuleName= str(appGuid) + rulename
asQuery="select * from AppServer "
rules[0].setMQLRuleName(RuleName)
rules[0].setFunctionalGroupName("App Servers")
rules[0].setMQLQuery(asQuery)

# création d'un nom de modèle d'application requis par TADDM, tel que
AA0A20EE5BBD336481279CA664FB380A:TADDM - Production
appTemplateName= str(appGuid) + ":" + BAname
appTemplate = api.createAppTemplate(appTemplateName, "APPLICATION", True,
jarray.array(rules,MQLRuleClass));
```

Exemple : Liste des applications métier

L'exemple suivant répertorie les applications métier :

```
query = "select * from Application"
data = api.executeQuery(query, None, None)
while (data.next()):
 print data.getXML(4)
```

Exemple : Suppression d'un modèle d'application métier

L'exemple suivant supprime un modèle d'application, les règles MQL et l'application métier associée.

```
# Obtenir un modèle d'application
appTemplate = api.getAppTemplate(nameBA, 0)

# Obtenir les règles du modèle d'application
rules = appTemplate.getMQLRules()

# Supprimer un modèle d'application -> supprime uniquement l'objet AppTemplate
api.removeAppTemplate(appTemplate.getAppTemplateName(), appTemplate.getAppTemplateType())

# Supprimer toutes les règles
pour les règles imbriquées :
rule = api.find(rule.getGuid(), 1, None)
api.removeMQLRule(rule.getMQLRuleName())

# Supprimer une application métier
businessApp = api.find("Select * from Application where name =='" +
appTemplate.getAppTemplateName()+ "'", False, None, None)
api.delete(businessApp, None)
```

Gestion des modèles de groupement :

Les méthodes de modèle de groupement permettent la création, la modification, la suppression et la récupération des modèles de groupement conjointement avec leurs sélecteurs.

Pour plus d'informations, voir la rubrique *Gestion des modèles de groupement à l'aide d'une API Java* dans le *Guide d'utilisation* de TADDM.

Utilisation de l'interface de programme d'application SOAP

L'interface de programme d'application Simple Object Access Protocol (SOAP) présente les éléments de l'interface de programme d'application TADDM comme service Web.

L'interface de programme d'application SOAP vous permet de développer des applications dans de nombreux environnements de développement et systèmes d'exploitation prenant en charge l'intégration aux applications de gestion y compris les gestionnaires de processus ITSM.

L'interface de programme d'application SOAP permet de contrôler le processus de reconnaissance et les aspects du Common Data Model y compris l'accès aux données de modèle obtenues. L'interface de programme d'application SOAP délègue les demandes à l'interface de programme d'application Java. L'interface de programme d'application Javapermet de créer des applications qui permettent d'ajouter, de mettre à jour et de supprimer les objets de modèle. Vous pouvez lancer une requête sur des objets de modèle de requête par nom de classe ou numéro ID objet.

Vous pouvez utiliser la commande de l'interface de programme d'application SOAP pour créer des applications permettant d'ajouter, de mettre à jour et de supprimer les objets de modèle. SOAP permet d'de lancer une requête sur des objets de modèle par nom de classe ou numéro ID objet. Cette interface vous permet également d'examiner l'historique des changements et de gérer les versions.

Récapitulatif de demande

L'interface de programme d'application SOAP offre un accès aux mappes d'application TADDM, y compris aux applications reconnues, à leurs composants et à leurs configurations.

L'interface de programme d'application SOAP se résume à l'aide des catégories suivantes, qui sont expliquées en détails dans leurs chapitres respectifs :

- Demandes de session
- Demandes de reconnaissance
- Gestion du modèle et des métadonnées
- Demandes de recherche
- Demandes d'historique des changements
- Gestion des versions

Demandes de session :

Les demandes de session vous permettent de gérer les sessions avec le serveur TADDM.

Les demandes de session permettent de se connecter au serveur TADDM et de s'en déconnecter. Le tableau 24, à la page 88 décrit les demandes de session que vous pouvez utiliser.

Tableau 24. Demandes de session

Opération	Entrée	Sortie
login (Connecte au serveur TADDM)	loginRequest **utilisateur** Le nom d'utilisateur enregistré avec le serveur TADDM **mot de passe** Correspond au mot de passe associé à l'utilisateur **hôte** Indique le nom de l'hôte, que ce soit un nom ou une adresse IP (à l'aide de la notation décimale) **port** Indique le numéro de port du serveur	loginReponse
logout (Se déconnecter du serveur)	logoutRequest	logoutReponse

Exemple

L'exemple suivant montre une demande XML de connexion :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:login soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <ns1:arg0 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">smartoperator</ns1:arg0>
      <ns1:arg1 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">foobar</ns1:arg1>
      <ns1:arg2 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">localhost</ns1:arg2>
      </ns1:login>
   </soapenv:Body>
</soapenv:Envelope>
```

L'exemple suivant montre la réponse XML de connexion :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:loginResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <loginReturn xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">1149902064172</loginReturn>
      </ns1:loginResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Demandes de reconnaissance :

Les méthodes de reconnaissance vous permettent de gérer les exécutions de reconnaissance.

Les demandes de reconnaissance permettent de démarrer et d'arrêter les opérations de reconnaissance, obtenir l'état d'une reconnaissance et de supprimer tous les éléments de reconnaissance de la topologie. Le tableau 25 décrit les demandes de reconnaissance que vous pouvez utiliser.

Tableau 25. Demandes de reconnaissance

Opération	Entrée	Sortie
abortDiscovery (Interrompre la reconnaissance qui est en cours)	abortDiscoveryRequest	abortDiscoveryResponse
clearTopology (Effacer tous les éléments de reconnaissance et les relations de la topologie)	clearTopologyRequest	clearTopologyResponse
getStatus (Obtenir l'état de l'opération de reconnaissance en cours)	getStatusRequest	getStatusResponse getStatusReturn : Représentation de la chaîne de l'état de l'opération de reconnaissance en cours
rebuildTopology (Reconstruire la topologie, y compris les dépendances et les relations)	rebuildTopologyRequest	rebuildTopologyResponse
startDiscovery (Démarrer une reconnaissance à l'aide de la portée indiquée)	startDiscoveryRequest **portée** La portée de la reconnaissance : nom de l'ensemble de portées ou du groupe de portées. **nomExécution** Désigne le nom de l'opération de reconnaissance.	startDiscoveryResponse
startDiscoveryRetID (Démarrer une reconnaissance à l'aide de la portée indiquée)	startDiscoveryRequest **portée** La portée de la reconnaissance : nom de l'ensemble de portées ou du groupe de portées. **nomExécution** Désigne le nom de l'opération de reconnaissance.	startDiscoveryResponse ID exécution

Gestion du modèle et des métadonnées :

Les demandes de modèle et de métadonnées vous permettent de gérer les métadonnées de demandes et les objets du Common Data Model. Les demandes de modèles permettent d'insérer, d'importer et d'exporter des objets dans le Common Data Model. Les demandes de métadonnées permettent d'obtenir tous les noms de classe du modèle qui peuvent être utilisés dans le langage de la requête.

Le tableau 26 décrit les demandes de modèle et de métadonnées que vous pouvez utiliser.

Tableau 26. Demandes de modèle et de métadonnées

Opération	Entrée	Sortie
exportData (Exporte tous les objets de niveau supérieur de la base de données TADDM vers un répertoire spécifié au format XML)	exportDataRequest **directoryToWriteTo** Désigne le nom du répertoire vers lequel les données doivent être exportées **maxfilesize** Indique la taille de fichier maximale à exporter **mssGuid** Désigne l'identificateur global unique du système logiciel de gestion	exportDataResponse
exportDataUsingMssName (Exporte tous les objets de niveau supérieur de la base de données TADDM vers un répertoire, au format XML, en spécifiant le système logiciel de gestion à l'aide d'un nom)	exportDataUsingMss NameRequest **directoryToWriteTo** Désigne le nom du répertoire vers lequel les données doivent être exportées **maxfilesize** Indique la taille de fichier maximale à exporter **mssGuid** Désigne l'identificateur global unique du système logiciel de gestion	exportDataUsingMss NameResponse
getClassNames (Obtenir tous les noms de classe du modèle qui peuvent être utilisés dans le langage de la requête)	getClassNamesRequest	getClassNamesResponse getClassNamesReturn : grappe de paires de nom qualifié complet/abrégé de la classe de modèle

Tableau 26. Demandes de modèle et de métadonnées (suite)

Opération	Entrée	Sortie
importData (Convertit les données XML de la source spécifiée dans les objets de modèle et met à jour la base de données TADDM)	importDataRequest **source** Désigne la source depuis laquelle les données doivent être importées **rebuildTopo** Désigne un booléen pour reconstruire la topologie **mssGuid** Désigne l'identificateur global unique du système logiciel de gestion	importDataResponse
importDataUsingMssName (Convertit les données XML de la source spécifiée dans les objets de modèle et met à jour la base de données TADDM, en spécifiant le système logiciel de gestion par son nom)	importDataUsingMss NameRequest **source** Désigne la source depuis laquelle les données doivent être importées **rebuildTopo** Désigne un booléen pour reconstruire la topologie **mssName** Il s'agit du nom du système logiciel de gestion	importDataUsingMss NameResponse
insert (Insère ou met à jour l'objet de modèle, spécifié au format XML, dans la base de données TADDM)	insertRequest **xml** Désigne l'objet de modèle au format XML **mssGuid** Désigne l'identificateur global unique du système logiciel de gestion	insertResponse
insertUsingMssName (Insérer l'objet de modèle, indiqué au format XML, spécifiant le système logiciel de gestion par son nom)	insertUsingMss NameRequest **xml** Désigne l'objet de modèle au format XML **mssName** Il s'agit du nom du système logiciel de gestion	insertUsingMss NameResponse

Exemple

L'exemple suivant montre une demande XML getClassNames :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getClassNames soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost"/>
   </soapenv:Body>
</soapenv:Envelope>
```

L'exemple suivant montre la réponse XML getClassNames :

```
GETCLASSNAME (response):
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:getClassNamesResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <getClassNamesReturn xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">
      AbstractResource,
      com.collation.platform.model.topology.process.AbstractResource,
      Accepts,
      com.collation.platform.model.topology.relation.Accepts,
       .
       .
       .
      </ns1:getClassNamesResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Demandes de recherche :

Les demandes de recherche vous permettent d'accéder aux objets du Common Data Model.

Les demandes de recherche vous permettent de renvoyer des objets de modèle qui répondent à un critère spécifique ou de renvoyer des informations sur des éléments gérés spécifiques. Elles permettent également de renvoyer les objets qui ont changé au cours d'une période spécifiée. Le tableau 27, à la page 93 décrit les opérations de recherche que vous pouvez exécuter.

Tableau 27. Demandes de recherche

Opération	Entrée	Sortie
find (Exécuter une requête pour un élément de configuration spécifié à l'aide du langage MQL)	findRequest **requête** Désigne la chaîne de requête **profondeur** Indique le niveau de l'arborescence de résultat à construire **retrait de ligne** Désigne le retrait de ligne à utiliser pour le fichier XML obtenu **mssGuid** Désigne l'identificateur global unique du système logiciel de gestion	findResponse findReturn : représentation XML des résultats de la requête
findBasedOnChange (Rechercher les objets qui ont été modifiés dans la période indiquée pour un type de changement donné)	findBasedOnChangeRequest **racine** Indique l'objet de modèle qui va servir de racine pour la sortie XML obtenue. **profondeur** Indique le niveau de l'arborescence de résultat à construire **retrait de ligne** Désigne le retrait de ligne à utiliser pour le fichier XML obtenu **début** Indique l'heure de début de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT **fin** Indique l'heure de fin de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT **typeChangement** Identifie le type de changement.	findBasedOnChangeResponse findBasedOnChangeReturn : représentation XML des résultats de la requête

Tableau 27. Demandes de recherche (suite)

Opération	Entrée	Sortie
findUsingGuid (Rechercher à l'aide de l'identificateur global unique de l'objet de modèle)	findUsingGuidRequest **identificateur global unique** Désigne l'identificateur global unique sur lequel la recherche doit être exécutée **profondeur** Indique le niveau de l'arborescence de résultat à construire **retrait de ligne** Désigne le retrait de ligne à utiliser pour le fichier XML obtenu	findUsingGuidResponse findUsingGuidReturn : représentation XML des résultats de la requête
findUsingMssName (Rechercher à l'aide du langage MQL, indiquant le système logiciel de gestion par son nom)	findUsingMssNameRequest **requête** Désigne la chaîne de requête **profondeur** Indique le niveau de l'arborescence de résultat à construire **retrait de ligne** Désigne le retrait de ligne à utiliser pour le fichier XML obtenu **mssName** Il s'agit du nom du système logiciel de gestion	findUsingMssNameResponse findUsingMssNameReturn : représentation XML des résultats de la requête

Exemple

L'exemple suivant montre une demande XML de recherche :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:find soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
      <ns1:arg0 xsi:type="soapenc:string" xmlns:soapenc=
         "http://schemas.xmlsoap.org/soap/encoding/">ComputerSystem</ns1:arg0>
      <ns1:arg1 href="#id0"/>
      <ns1:arg2 href="#id1"/>
      <ns1:arg3 xsi:nil="true"/>
    </ns1:find>
    <multiRef id="id1" soapenc:root="0" soapenv:encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">4</multiRef>
    <multiRef id="id0" soapenc:root="0" soapenv:encodingStyle=
       "http://schemas.xmlsoap.org/soap/encoding/" xsi:type="soapenc:int"
       xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">2</multiRef>
   </soapenv:Body>
</soapenv:Envelope>
```

L'exemple suivant montre la réponse XML de la recherche :

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi=
"http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Body>
      <ns1:findResponse soapenv:encodingStyle=
         "http://schemas.xmlsoap.org/soap/encoding/" xmlns:ns1="http://localhost">
         <findReturn xsi:type="soapenc:string" xmlns:soapenc=
            "http://schemas.xmlsoap.org/soap/encoding/">
         &lt;?xml version=&quot;1.0&quot; encoding=&quot;ISO-8859-1&quot;?&gt;
         &lt;results
         xmlns=&quot;urn:www-collation-com:1.0&quot;
          .
          .
          .
         &lt;architecture&gt;Intel&lt;/architecture&gt;
         &lt;/ComputerSystem&gt;
         &lt;/results&gt;
         </findReturn>
      </ns1:findResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

Demandes d'historique des changements :

Les demandes d'historique des changements permettent de déterminer l'historique des changements dans le Common Data Model.

Les demandes d'historique des changements vous permettent d'extraire l'historique des changements des éléments gérés dans le Common Data Model. Le tableau 28 décrit les demandes d'historique des changements que vous pouvez utiliser.

Tableau 28. Demandes d'historique des changements

Opération	Entrée	Sortie
getChangeHistory (Obtenir l'historique des changements pour la période de début et de fin à l'aide de l'identificateur global unique spécifié)	getChangeHistoryRequest **identificateur global unique** Correspond à l'identificateur global unique de l'objet pour lequel l'historique des changements est requis **début** Indique l'heure de début de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT **fin** Indique l'heure de fin de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT	getChangeHistoryResponse getChangeHistoryReturn : représentation XML d'une liste d'objets ChangeHistory

Tableau 28. Demandes d'historique des changements (suite)

Opération	Entrée	Sortie
getChangeHistory (Obtenir l'historique des changements pour la période de début et de fin pour plusieurs identificateurs globaux uniques)	getChangeHistoryRequest1 **identificateurs globaux uniques** Présente une liste d'identificateurs globaux uniques séparés par des virgules des objets pour lesquels l'historique des changements est requis **début** Indique l'heure de début de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT **fin** Indique l'heure de fin de la période de changement, indiquée en millisecondes à partir du 1er janvier, 1970, 00:00:00 GMT	getChangeHistoryResponse1 getChangeHistoryReturn : représentation XML d'une liste d'objets ChangeHistory

Gestion des versions :

Les méthodes de version vous permettent de gérer les versions de données de la base de données TADDM. Les demandes de version permettent de créer des images instantanées convenues des données actuelles de la base de données TADDM, de supprimer des versions et d'établir la liste des versions définies disponibles.

Le tableau 29 décrit les demandes de version que vous pouvez utiliser.

Tableau 29. Demandes de version

Opération	Entrée	Sortie
createVersion (Crée une image instantanée convenue des données actuelles de la base de données TADDM)	createVersionRequest **nom** Désigne le nom de la version. **description** Il s'agit de la description de la nouvelle version	createVersionResponse

Tableau 29. Demandes de version (suite)

Opération	Entrée	Sortie
createEmptyVersion (Crée une version vide, sans aucune donnée, dans la base de données TADDM)	createEmptyVersionRequest **nom** Désigne le nom de la version. **description** Il s'agit de la description de la nouvelle version	createEmptyVersionResponse
deleteVersion (Supprime une version de la base de données TADDM)	deleteVersionRequest **IDversion** Correspond à l'identificateur de la version	deleteVersionResponse
deleteVersionUsingName (Supprime une version, identifiée par son nom, de la base de données TADDM)	deleteVersionUsing NameRequest **nomVersion** Désigne le nom de la version.	deleteVersionUsing NameResponse
getAllVersions (Obtient les noms de toutes les versions de données définies de la base de données TADDM)	getAllVersionsRequest	getAllVersionsResponse

Développement d'applications à l'aide de l'interface de programme d'application REST

Vous pouvez utiliser l'interface de programme d'application REST TADDM pour développer des applications qui accèdent à des ressources TADDM sélectionnées en suivant les principes HTTP et REST.

Présentation de l'interface de programme d'application REST

L'interface de programme d'application REST expose un sous-ensemble des fonctions de l'interface de programme d'application Java aux clients et aux navigateurs Web utilisant HTTP. A l'aide des ressources REST, vous pouvez développer des applications pour tout système d'exploitation et toute langue prenant en charge les appels HTTP.

Les ressources REST exposent les fonctions TADDM que vous pouvez utiliser pour rechercher des objets de modèle par requêtes sur nom de classe, identificateur global unique ou langage MQL. Vous pouvez également créer, supprimer et mettre à jour des objets modèles, ou gérer le processus de reconnaissance TADDM. Toutes ces fonctions utilisent des interfaces HTTP standard et prennent en charge le format JSON ou XML pour les données d'entrée et de sortie.

Les composants du serveur REST sont installés dans le répertoire `$COLLATION_HOME/deploy-tomcat` (TADDM 7.3.0) ou `$COLLATION_HOME/apps` (TADDM versions 7.3.0.1 et ultérieures) du serveur TADDM et démarrent automatiquement au lancement du serveur TADDM. Les services REST sont disponibles sur les mêmes ports TCP/IP qu'utilise l'interface Web administrative de TADDM. (Les ports par défaut sont 9430 pour HTTP et 9431 pour HTTPS.)

L'interface de programme d'application REST utilise l'authentification de base HTTP pour la transmission de l'ID utilisateur et du mot de passe à l'aide du codage MIME Base64. Comme chaque requête est sans état, chaque appel à l'interface de programme d'application REST doit inclure l'en-tête de l'autorisation HTTP. Pour les connexions sécurisées, utilisez une connexion HTTPS.

Les paramètres des appels REST sont spécifiés à l'aide de la notation de chaîne de requête spécifiée :

`http://url_ressource?paramètre=valeur&paramètre=valeur...`

Si vous spécifiez une valeur de paramètre non valide, le serveur TADDM ignore cette valeur et utilise la valeur par défaut, si cette dernière peut être déterminée. (Par exemple, si vous spécifiez `fetchSize=-2`, le serveur utilise la valeur **`fetchSize`** 1.) En l'absence de spécification possible d'une valeur par défaut, la requête échoue.

Appels REST à l'aide d'un navigateur Web

Vous pouvez effectuer de nombreux appels d'API REST en entrant les adresses URL appropriées dans un navigateur Web.

Avant de commencer

Pour accéder aux interfaces REST en toute sécurité à l'aide d'une connexion HTTPS, vous devez commencer par configurer votre navigateur pour qu'il accepte les connexions TLS (Transport Layer Security) (TLS) 1.0.

Pourquoi et quand exécuter cette tâche

L'interface de programme d'application REST utilise plusieurs méthodes HTTP pour exécuter diverses actions au niveau des ressources REST. Tout appel d'API REST qui utilise la méthode HTTP GET peut être soumis à l'aide d'un navigateur Web du type Microsoft Internet Explorer ou Mozilla Firefox.

Procédure

Pour accéder à un appel REST via un navigateur, procédez comme suit :

Entrez l'adresse URL appropriée via HTTP ou HTTPS.

- Dans cet exemple, une requête spécifiée avec le langage MQL (Model Query Language) est soumise via HTTP :

  ```
  http://yourhost.com:9430/rest/model/MQLQuery?query=select%20name%20from%
  20ComputerSystem%20where%20signature%20starts-with%20'M&Y'&feed=xml&fetc
  hSize=100&position=1
  ```

- Dans cet exemple, une requête ComputerSystem est soumise via HTTPS :

  ```
  https://yourhost.com:9431/rest/model/ComputerSystem?depth=1
  ```

Lors de votre premier accès à l'API REST TADDM via un navigateur, une page de connexion vous invite à fournir un ID utilisateur TADDM et un mot de passe valides.

Appels REST dans une application Java

Vous pouvez avoir recours à des méthodes Java standard pour accéder à l'interface de programme d'application REST TADDM.

Avant de commencer

Pour accéder aux interfaces REST en toute sécurité à l'aide d'une connexion HTTPS, vous devez commencer par copier le certificat de sécurité jssecacerts.cert sur le système client. Ce fichier se trouve dans le répertoire `$COLLATION_HOME/etc` sur le serveur TADDM.

Procédure

L'exemple suivant montre comment accéder à l'interface de programme d'application REST à partir d'un programme Java.

Pour accéder à l'interface de programme d'application REST à partir d'un programmeJava, servez-vous des méthodes Java standard pour les communications HTTP. Cet exemple montre comment accéder à l'interface de programme d'application REST à l'aide d'une connexion HTTPS sécurisée :

```
HostnameVerifier hv = new HostnameVerifier() {
             public boolean verify(String urlHostName, SSLSession session) {
                System.out.println("Warning: URL Host: "+urlHostName+" vs. "
                  +session.getPeerHost());
                return true;
             }
         };

         // set this property to the location of the cert file
         System.setProperty("javax.net.ssl.trustStore", "jssecacerts.cert");

         HttpsURLConnection.setDefaultHostnameVerifier(hv);
         URL url = new
           URL("https://cab.tivlab.austin.ibm.com:9431/rest/model/"+
               "Repository?depth=1&feed=json");
         HttpsURLConnection urlConn = (HttpsURLConnection) url.openConnection();

         System.out.println("sending request...");
         urlConn.setRequestMethod("GET");
         urlConn.setAllowUserInteraction(false); // no user interaction
         urlConn.setDoOutput(true); // want to send
         urlConn.setRequestProperty( "Content-type", "text/xml" );
         urlConn.setRequestProperty( "accept", "text/xml" );
         urlConn.setRequestProperty( "authorization", "Basic " +
             encode("administrator:collation"));
         Map headerFields = urlConn.getHeaderFields();
         System.out.println("header fields are: " + headerFields);

         int rspCode = urlConn.getResponseCode();
         if (rspCode == 200) {
             InputStream ist = urlConn.getInputStream();
             InputStreamReader isr = new InputStreamReader(ist);
             BufferedReader br = new BufferedReader(isr);

             String nextLine = br.readLine();
             while (nextLine != null) {
                 System.out.println(nextLine);
                 nextLine = br.readLine();
             }
         }
```

Analyse des résultats d'une requête REST

Pour analyser les résultats d'une requête REST dans une application Java, vous avez le choix entre les méthodes XPath ou JXPath standard.

Avant de commencer

Vérifiez que vous avez accès à une bibliothèque XPath pour l'analyse des données XML ou à une bibliothèque JXPath pour l'analyse des données JSON. La prise en charge XPath est incluse dans le kit de développement de logiciels IBM Java Technology Edition version 5 et ultérieure. La prise en charge JXPath est incluse dans le logiciel SDK TADDM.

Procédure

L'exemple suivant montre comment utiliser ces fonctions pour analyser la sortie des appels REST TADDM.

Vous pouvez ensuite utiliser ces fonctions pour analyser la sortie des appels REST TADDM. L'exemple suivant montre comment renvoyer la valeur serviceName de chacun des services installés (installedServices) pour un système d'exploitation, en utilisant JXPath pour analyser les données de sortie JSON.

Remarque : La classe JSONArray fait partie du module json-simple, qui n'est pas inclus avec le logiciel SDK TADDM. Pour télécharger ce module, rendez-vous à la page http://code.google.com/p/json-simple/.

```
//queryResult contains the results from a TADDM Query
JSONArray arrayObject = (JSONArray) JSONValue.parse(queryResult);
   JXPathContext context2 = JXPathContext.newContext(arrayObject);
   Iterator names = context2.iterate("//installedServices/serviceName");
   while(names.hasNext()) {
    String serviceName = (String) names.next();
    System.out.println("service name is: " + serviceName);
  }
```

Débogage d'applications REST

S votre application rencontre des erreurs d'accès à l'interface de programme d'application REST, plusieurs techniques sont à votre disposition pour vous permettre de déterminer la nature du problème.

Pourquoi et quand exécuter cette tâche

L'API REST utilise plusieurs mécanismes pour indiquer les résultats des appels et erreurs REST se produisant lors du traitement. Servez-vous de ces méthodes pour déboguer une application REST.

Procédure

Utilisez les méthodes suivantes pour déboguer une application REST.

- Vérifiez le code réponse HTTP. Les codes réponse courants sont les suivants :

  200 La requête a abouti.

  400 Les données d'entrée n'étaient pas valides.

  409 Le serveur a détecté un conflit, du type tentative d'ajout d'un objet existant déjà.

  500 Une erreur serveur s'est produite.
- Pour plus d'informations, consultez le message de réponse dans l'en-tête HTTP.
- Vérifiez la présence de messages pertinents dans les fichiers journaux. Il est possible que les messages apparaissent dans les fichiers suivants :
  - `$COLLATION_HOME/log/tomcat.log` (TADDM 7.3.0)

– $COLLATION_HOME/log/wlp.log (TADDM versions 7.3.0.1 et ultérieures)
– $COLLATION_HOME/log/services/ApiServer.log

Requête sur des objets de modèle à l'aide de l'interface de programme d'application REST

Vous avez le choix entre deux méthodes pour lancer une requête sur un objet de modèle en utilisant l'interface de programme d'application REST.

Procédure

Pour utiliser l'interface de programme d'application REST pour lancer une requête sur des objets de modèle, effectuez l'une des deux étapes suivantes :

- Pour lancer une requête sur des objets de modèle en spécifiant leur classe, servez-vous de la ressource de classe d'objet de modèle. Vous pouvez utiliser cette ressource pour les requêtes d'informations sur des objets de modèle d'une classe donnée, en incluant en option des valeurs d'attribut spécifiées. Cette ressource constitue une méthode simple de requête pour des objets d'une classe donnée. Dans cet exemple, le cinquième objet ComputerSystem dont l'attribut OSRunning est défini sur `linux` fait l'objet d'une requête :

  ```
  http://example.com:9430/rest/model/ComputerSystem?depth=2&feed=xml&OSRu
  nning.OSName=Linux&position=5
  ```

- Pour lancer une requête sur des objets de modèle à l'aide du langage MQL (Model Query Language), servez-vous de la ressource du service de requête MQL. Cette ressource accepte toute requête pouvant être exprimée via le langage MQL et est plus souple que la ressource de classe d'objet de modèle. Dans cet exemple, la requête MQL `select displayName,OSRunning.OSName from ComputerSystem` s'applique aux données de l'objet de modèle.

  ```
  http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,
  OSRunning.OSName%20from%20ComputerSystem&position=2&fetchSize=2&feed=xm
  l&depth=2&position=4
  ```

Ajout d'objets de modèle à l'aide de l'interface de programme d'application REST

Vous avez le choix entre deux méthodes pour ajouter un nouvel objet de modèle en utilisant l'interface de programme d'application REST.

Pourquoi et quand exécuter cette tâche

La ressource du service de mise à jour d'objet modèle prend en charge la création de nouveaux objets de modèle à l'aide de la méthode HTTP POST ou PUT, suivant que vous voulez ou non permettre la modification d'un objet existant.

Dans l'un ou l'autre cas, vous devez commencer par décrire les nouvelles données d'objet en utilisant le format JSON ou XML ; le serveur détecte automatiquement le format utilisé. Si vous devez spécifier un objet de modèle complexe, il peut être intéressant de commencer par rechercher les métadonnées de classe à l'aide du service adéquat. Les résultats de cette requête fournissent les noms d'attribut corrects de l'objet, que vous pouvez alors utiliser pour spécifier les nouvelles données au format XML ou JSON.

Dans certains cas, il se peut que vous deviez ajouter un objet de modèle incluant un deuxième objet de modèle comme attribut, avec l'attribut `parent` requis sur l'objet enfant. Vous devez alors commencer par déterminer l'identificateur global unique de l'objet parent pour pouvoir définir l'attribut `parent` de l'objet enfant. Pour ce faire, deux méthodes sont possibles :

- Créez d'abord l'objet parent, sans l'objet enfant, pour pouvoir déterminer l'identificateur global unique du parent. Créez ensuite l'objet enfant, en spécifiant l'identificateur global unique de l'objet parent.
- Créez les deux objets dans une même requête. Pour ce faire, vous devez définir l'identificateur global unique de l'objet parent en lui donnant une valeur unique dans le document JSON ou XML, puis spécifier cette même valeur pour l'attribut `parent` de l'objet enfant. Dans cet exemple JSON, l'ID `cs1` est utilisé comme identificateur global unique de l'objet parent :

  ```
  [{"signature":"JsonRestExample1","_class":"LinuxUnitaryComputerSystem","
  numCPUs":2,"guid":"cs1","OSRunning":{"_class":"Linux","parent":"cs1","na
  me":"Linux","description":"Created by sample code"}}]
  ```

  Pour d'autres exemples, reportez-vous aux exemples de programmes dans le répertoire `$COLLATION_HOME/sdk/examples/rest`.

Procédure

Utilisez l'une des deux méthodes suivantes :

- Utilisez le service de mise à jour d'objet de modèle et la méthode HTTP POST, en faisant passer les données du nouvel objet dans le corps de la requête. Cette méthode aboutit uniquement si l'objet spécifié n'existe pas préalablement. S'il existe déjà, la requête échoue. Choisissez cette méthode si vous voulez créer un nouvel objet sans pour cela modifier d'objets existants.
- Utilisez le service de mise à jour d'objet de modèle et la méthode HTTP PUT, en faisant passer les données du nouvel objet dans le corps de la requête. Cette méthode crée l'objet spécifié, si celui-ci n'existe pas déjà ; s'il existe déjà, il est modifié à l'aide des nouvelles données. Choisissez cette méthode si vous voulez vous assurer que l'objet spécifié se trouve dans la base de données TADDM, qu'il existe ou non préalablement.

Mise à jour d'objets de modèle à l'aide de l'interface de programme d'application REST

Vous avez le choix entre deux méthodes pour mettre à jour un objet de modèle existant en utilisant l'interface de programme d'application REST.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser une ressource d'objet de modèle ou le service de mise à jour d'objet de modèle pour mettre à jour un objet de modèle existant, suivant que vous voulez ou non permettre la création de nouveaux objets.

Dans l'un ou l'autre cas, vous devez commencer par décrire les nouvelles données d'objet en utilisant le format JSON ou XML ; le serveur détecte automatiquement le format utilisé. Si vous devez spécifier un objet de modèle complexe, il peut être intéressant de commencer par rechercher les métadonnées de classe à l'aide du service adéquat. Les résultats de cette requête fournissent les noms d'attribut corrects de l'objet, que vous pouvez alors utiliser pour spécifier les nouvelles données au format XML ou JSON.

Procédure

Pour utiliser l'interface de programme d'application REST pour mettre à jour des objets de modèle existants, effectuez l'une des deux méthodes suivantes :

- Utilisez la ressource d'objet de modèle représentant l'objet existant ainsi que la méthode HTTP PUT, en faisant passer les données du nouvel objet dans le corps

de la requête. Cette ressource est disponible uniquement si l'objet spécifié n'existe pas préalablement. Dans le cas contraire, la requête échoue. Choisissez cette méthode si vous voulez modifier un objet existant, mais que vous ne voulez pas ajouter l'objet s'il n'existe pas déjà.

- Utilisez le service de mise à jour d'objet de modèle et la méthode HTTP PUT, en faisant passer les données du nouvel objet dans le corps de la requête. Cette méthode met à jour l'objet avec les nouvelles données ; si l'objet n'existe pas déjà, il est créé. Choisissez cette méthode si vous voulez vous assurer qu'un objet contenant les données spécifiées se trouve dans la base de données TADDM, qu'il existe ou non préalablement.

Suppression d'objets de modèle à l'aide de l'interface de programme d'application REST

Vous avez le choix entre deux méthodes pour supprimer un objet de modèle en utilisant l'interface de programme d'application REST.

Pourquoi et quand exécuter cette tâche

Un seul objet peut être supprimé lors d'une seule requête. Si l'objet spécifié n'existe pas, aucune erreur n'est renvoyée.

Procédure

Pour supprimer un objet de modèle, effectuez l'une des deux méthodes suivantes :

- Pour supprimer un objet en spécifiant son identificateur global unique, utilisez la ressource d'objet de modèle correspondante, en spécifiant l'identificateur de l'objet à supprimer dans le cadre de l'adresse URL et en utilisant la méthode HTTP DELETE. Si vous utilisez cette méthode, aucune donnée n'est requise dans le corps de la requête HTTP. Dans cet exemple, soumis via la méthode HTTP DELETE, un objet est supprimé à l'aide de la ressource d'ob jet de modèle :

  ```
  http://example.com:9430/rest/model/ModelObject/1D646C44FDEB3857B40B98BD
  F9C0F407?mssGuid=CF5EBF574E7F382289B3F35FB5776628
  ```

- Utilisez le service de mise à jour d'objet de modèle avec le paramètre **delete** et la méthode HTTP POST, en spécifiant l'objet à supprimer dans le corps de la requête HTTP. Seul l'identificateur global unique de l'objet à supprimer est requis dans les données d'entrée, même s'il est possible de spécifier l'objet entier. Dans cet exemple, soumis via la méthode HTTP POST, l'objet spécifié dans le corps de la requête est supprimé :

  ```
  http://example.com:9430/rest/model/ModelObject?delete=true
  ```

Gestion des modèles de groupement à l'aide de l'interface de programme d'application REST

Les modèles de groupement peuvent être gérés sans passer par une interface utilisateur mais à l'aide de l'interface de programme d'application REST.

L'interface de programme d'application REST pour les modèles de groupement est accessible à l'aide de justificatifs d'identité standard. Une authentification WWW basique est nécessaire. Si vous souhaitez accéder à l'interface de programme d'application REST via un navigateur Web avec une session utilisateur déjà ouverte, vous n'avez pas besoin de vous reconnecter.

Les données peuvent être envoyées et reçues sous deux formats : JSON et XML (application/json et application/xml). Le chemin d'accès à ce service est le suivant : `/cdm/api/groupingpatterns`.

La liste suivante répertorie les méthodes que vous pouvez utiliser et les actions exécutées par celles-ci :

- `/cdm/api/groupingpatterns :: method GET` - returns all grouping patterns.
- `/cdm/api/groupingpatterns/{GUID} :: method GET` - returns grouping patterns with the given GUID.
- `/cdm/api/groupingpatterns :: method POST :: load in JSON or XML format` - creates new grouping patterns.
- `/cdm/api/groupingpatterns/{GUID} :: method DELETE` - deletes grouping patterns with the given GUID.
- `/cdm/api/groupingpatterns/ :: method PUT :: load in JSON or XML format` - updates grouping patterns.

Exemple de chargement au format XML :

```
<groupingPattern name="GroupingPattern1" hierarchyType="ACCESS_COLLECTION" active="true">
    <selector name="Selector1" lowerDown="include" lowerUp="include" higherDown="include"
 higherUp="include" GroupingNameExpression="GroupingNameExpression"
useTraversalTemplate="false" type="MQL">
        <query>ComputerSystem where guid == '00000000000000000000000000000000'</query>
    </selector>
    <description>GroupingPattern1Description</description>
</groupingPattern>
```

Gestion de reconnaissances à l'aide de l'interface de programme d'application REST

Vous pouvez utiliser l'interface de programme d'application REST pour démarrer la reconnaissance et pour gérer les reconnaissances, ainsi que les profils et les portées de reconnaissance.

Procédure

Pour utiliser l'API REST, effectuez une ou plusieurs des étapes suivantes :

- Pour démarrer une reconnaissance, utilisez la demande POST d'envoi de ressource du service de reconnaissance en indiquant un nom pour l'opération de reconnaissance. Vous pouvez utiliser cette ressource pour démarrer une reconnaissance avec ou sans profil. Par exemple, ce formulaire HTML démarre la reconnaissance nommée "TestRun2" en utilisant le profil spécifié pour la portée spécifiée.

  ```
  <form action="http://example.com:9430/rest/discovery/start/TestRun2" method="post">
       Profil : <input type="text" name="profile"><br>
       Portée : <input type="text" name="scope"><br>
       <input type="submit" value="Submit"></input>
  </form>
  ```

- Pour vérifier le statut de reconnaissance en cours, utilisez la ressource de statut de reconnaissance, en spécifiant le format XML ou JSON pour les données de sortie. Dans cet exemple, le statut de reconnaissance est vérifié dans le format JSON :

  ```
  http://example.com/rest/discovery/status?feed=json
  ```

- Pour extraire la liste des profils de reconnaissance définis, utilisez la ressource du service de profil de reconnaissance, en spécifiant le format XML ou JSON pour les données de sortie. Dans cet exemple, la liste des profils de reconnaissance est indiquée dans le format XML :

  ```
  http://example.com/rest/discovery/profiles?feed=xml
  ```

- Pour extraire les détails relatifs à un profil de reconnaissance défini, utilisez la ressource du profil de reconnaissance, en spécifiant le format XML ou JSON pour les données de sortie. Les détails relatifs au profil `Level 3 Discovery` sont extraits dans le format JSON :

```
http://example.com/rest/discovery/profile/Level%203%20Discovery?feed=json
```

- Pour extraire la liste des portées de reconnaissance définies, utilisez la ressource du service de portée de reconnaissance, en spécifiant le format XML ou JSON pour les données de sortie. Dans cet exemple, la liste des portées de reconnaissance est indiquée dans le format XML :

  ```
  http://example.com/rest/discovery/scopes?feed=xml
  ```

- Pour extraire les détails relatifs à une portée de reconnaissance définie, utilisez la ressource de portée de reconnaissance, en spécifiant le format XML ou JSON pour les données de sortie. Les détails relatifs à la portée `scope1` sont extraits dans le format JSON :

  ```
  http://example.com/rest/discovery/scope/scope1?feed=json
  ```

Référence de ressource REST

L'interface de programme d'application REST expose les ressources que vous pouvez utiliser pour lancer une requête, créer, mettre à jour et supprimer des objets modèles, ainsi que pour gérer des reconnaissances.

Classe d'objet de modèle :

La ressource de classe d'objet de modèle représente une classe d'objets de modèle définie par le Common Data Model (CDM).

Description

Utilisez cette ressource pour extraire des informations sur des objets de modèle en spécifiant une classe d'objet de modèle, et en incluant en option des valeurs d'attribut. Ce type de requête fournit un sous-ensemble d'informations disponibles via des requêtes MQL.

Servez-vous de la méthode HTTP GET pour envoyer une demande de requête MQL.

URL

scheme//*hostname*:*port*/rest/model/*model_object_class*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

model_object_class
Nom de classe de l'objet de modèle. Spécifiez le nom abrégé (comme ComputerSystem) ou le nom qualifié complet (comme com.collation.platform.model.topology.sys.ComputerSystem).

Méthodes HTTP

GET Recherche des objets de modèle.

Paramètres

cols=*value*
Liste (séparée par des virgules) de noms de colonne pour lesquels vous voulez que des données soient renvoyées. La valeur par défaut est le renvoi de données à partir de toutes les colonnes.

depth=*value*
Profondeur de la requête. La valeur par défaut est 1.

Remarque : Une requête avec une profondeur supérieure à 1 peut renvoyer un grand volume de résultats, entraînant de mauvaises performances de la mémoire sur le serveur TADDM. Pour éviter ceci, spécifiez `fetchSize=1` et utilisez des requêtes consécutives pour faire défiler les données position par position. Pour des exemples d'utilisation de cette technique, reportez-vous aux exemples de programme dans le répertoire `$COLLATION_HOME/sdk/examples/rest`.

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

fetchSize=*value*
Nombre maximal d'objets à renvoyer à partir de l'ensemble de résultats. La valeur par défaut est 1.

longClassName={true|false}
Indique si les noms de classe de l'objet de modèle doivent tous être spécifiés comme un nom qualifié complet (par exemple, com.collation.platforles noms m.model.topology.sys.ComputerSystem). La valeur doit être `true` ou `false`. Cette option est valide uniquement pour la sortie JSON. La valeur par défaut est `false`.

mssGuid=*value*
Valeur de l'identificateur global unique du système logiciel de gestion associé à l'objet. Ce paramètre est facultatif.

position=*value*
Position de départ dans l'ensemble de résultats des objets que vous voulez recevoir de la requête. La valeur par défaut est 1 (le premier objet de l'ensemble de résultats). Si vous spécifiez une position supérieur au nombre total d'objets présents dans l'ensemble de résultats, aucun objet n'est renvoyé.

attribute_name=attribute_value
Valeur et nom d'attribut (en option). Spécifiez cette option pour limiter la sortie de la requête aux objets correspondant à la valeur d'attribut spécifiée. Si vous spécifiez plusieurs attributs, seuls les objets correspondant aux valeurs d'attribut spécifiées sont renvoyées.

Résultats

Si la requête aboutit, le serveur renvoie le code retour HTTP 200, et les données du résultat de la requête au format JSON ou XML (comme spécifié par le paramètre **feed** ou l'en-tête HTTP Accept). Si la requête ne renvoie aucune donnée, l'ensemble de résultats est un tableau JSON ou un document XML vide, suivant le type feed spécifié.

L'en-tête TADDMQueryComplete des données renvoyées indique si tous les résultats disponibles de la requête ont été renvoyés : `true` indique que tous les résultats ont été renvoyés, et `false` que d'autres résultats sont disponibles. Vous pouvez contrôler les résultats reçus en réglant les valeurs des paramètres facultatifs `position` et `fetchSize`.

Exemple

Dans cet exemple, le cinquième objet ComputerSystem dont l'attribut OSRunning est défini sur `linux` fait l'objet d'une requête :

```
http://example.com:9430/rest/model/ComputerSystem?depth=2&feed=xml&OSRu
nning.OSName=Linux&position=5
```

Service de requête MQL :

La ressource de service de requête MQL extrait des données de l'objet de modèle sur la base de requêtes écrites en langage MQL.

Description

Servez-vous de cette ressource pour extraire des données d'un objet de modèle en utilisant des requêtes écrites en langage MQL. Le service de requête MQL peut fournir des informations plus détaillées que la ressource de classe d'objet de modèle.

URL

scheme//*hostname*:*port*/rest/model/MQLQuery

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

Méthodes HTTP

GET Recherche des objets de modèle.

Paramètres

depth=*value*
Profondeur de la requête. La valeur par défaut est 1.

Remarque : Une requête avec une profondeur supérieure à 1 peut renvoyer un grand volume de résultats, entraînant de mauvaises performances de la mémoire sur le serveur TADDM. Pour éviter ceci, spécifiez `fetchSize=1` et utilisez des requêtes consécutives pour faire défiler les données position par position. Pour des exemples d'utilisation de cette technique, reportez-vous aux exemples de programme dans le répertoire `$COLLATION_HOME/sdk/examples/rest`.

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

fetchSize=*value*
Nombre maximal d'objets à renvoyer à partir de l'ensemble de résultats. La valeur par défaut est 1.

longClassName={true|false}
Indique si les noms de classe de l'objet de modèle doivent tous être spécifiés comme un nom qualifié complet (par exemple, com.collation.platforles noms m.model.topology.sys.ComputerSystem). La valeur doit être `true` ou `false`. Cette option est valide uniquement pour la sortie JSON. La valeur par défaut est `false`.

mssGuid=*value*
Valeur de l'identificateur global unique du système logiciel de gestion associé à l'objet. Ce paramètre est facultatif.

position=*value*
Position de départ dans l'ensemble de résultats des objets que vous voulez recevoir de la requête. La valeur par défaut est 1 (le premier objet de l'ensemble de résultats). Si vous spécifiez une position supérieur au nombre total d'objets présents dans l'ensemble de résultats, aucun objet n'est renvoyé.

query=*value*
La chaîne de recherche, en notation MQL. Ce paramètre est obligatoire.

Remarque : Les requêtes portant sur un objet de modèle peuvent renvoyer un grand volume de données. Pour éviter tout problème de mémoire et de performances, sélectionnez uniquement les colonnes dont vous avez besoin.

Résultats

Si la requête aboutit, le serveur renvoie le code retour HTTP 200, et les données du résultat de la requête au format JSON ou XML (comme spécifié par le paramètre **feed** ou l'en-tête HTTP Accept). Si la requête ne renvoie aucune donnée, l'ensemble de résultats est un tableau JSON ou un document XML vide, suivant le type feed spécifié.

L'en-tête TADDMQueryComplete des données renvoyées indique si tous les résultats disponibles de la requête ont été renvoyés : `true` indique que tous les résultats ont été renvoyés, et `false` que d'autres résultats sont disponibles. Vous pouvez contrôler les résultats reçus en réglant les valeurs des paramètres facultatifs `position` et `fetchSize`.

Exemple

Dans cet exemple, la requête MQL `select displayName,OSRunning.OSName from ComputerSystem` s'applique aux données de l'objet de modèle.

```
http://example.com:9430/rest/model/MQLQuery?query=select%20displayName,
OSRunning.OSName%20from%20ComputerSystem&position=2&fetchSize=2&feed=xm
l&depth=2&position=4
```

Objet de modèle :

Une ressource d'objet de modèle représente une instance d'objet de modèle spécifique qui existe dans la base de données TADDM, identifiée par un identificateur global unique.

Description

Servez-vous de ce type de ressource pour rechercher, mettre à jour ou supprimer une instance d'objet de modèle identifiée par son identificateur global unique.

URL

scheme//*hostname*:*port*/rest/model/ModelObject/*guid*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

guid
Identificateur global unique d'une instance d'objet de modèle qui existe dans la base de données TADDM. Si vous effectuez la mise à niveau d'un objet, cet identificateur doit correspondre à celui qui est spécifié dans les données de l'objet JSON ou XML.

Méthodes HTTP

GET Recherche un objet de modèle.

PUT Met à jour un objet de modèle. Les données du nouvel objet doivent être spécifiées dans le corps de la requête HTTP au format JSON ou XML. (Le serveur détecte automatiquement le format des données d'entrée.)

DELETE
Supprime un objet de modèle.

Paramètres

depth=*value*
Profondeur de la requête. La valeur par défaut est 1. Ce paramètre n'est pas utilisé lors de la mise à jour ou de la suppression d'objets.

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Le paramètre **feed** n'est pas utilisé lors de la mise à jour ou de la suppression d'un objet.

longClassName={true|false}
Indique si les noms de classe de l'objet de modèle sont spécifiés comme un nom qualifié complet (par exemple, com.collation.platforles noms m.model.topology.sys.ComputerSystem). La valeur doit être `true` ou `false`. Cette option est valide uniquement pour la sortie JSON. La valeur par défaut est `false`.

mssGuid=*value*
Valeur de l'identificateur global unique du système logiciel de gestion associé à l'objet. Ce paramètre est facultatif.

Résultats

Si la requête aboutit, le serveur renvoie le code retour HTTP 200. Pour une requête, le serveur renvoie également les données du résultat au format JSON ou XML (comme spécifié par le paramètre **feed** ou l'en-tête HTTP Accept). Si la requête ne renvoie aucune donnée, l'ensemble de résultats est un tableau JSON ou un document XML vide, suivant le type feed spécifié.

Exemple

Dans cet exemple, un objet existant est recherché, mis à jour ou supprimé, suivant la méthode HTTP utilisée. (Pour mettre à jour un objet, le corps de la requête doit contenir les données de l'objet mis à jour.)

```
http://example.com:9430/rest/model/ModelObject/1D646C44FDEB3857B40B98BD
F9C0F407?mssGuid=CF5EBF574E7F382289B3F35FB5776628
```

Métadonnées de classe d'objet de modèle :

La ressource des métadonnées de classe d'objet de modèle représente les métadonnées décrivant les attributs d'une classe d'objet de modèle.

Description

Utilisez la ressource des métadonnées de classe d'objet de modèle pour rechercher des données sur les attributs d'une classe d'objet de modèle, notamment le numéro, type et nom de chaque attribut. Ces informations sont équivalentes à celles renvoyées par la méthode Java getMetaData().

Les métadonnées peuvent être renvoyées au format JSON ou XML.

URL

scheme//*hostname*:*port*/rest/model/meta/*model_object_class*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

model_object_class
Nom d'une classe d'objet Common Data Model (CDM).

Méthodes HTTP

GET Recherche les métadonnées de classe d'un objet.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Exemple

Dans cet exemple, les informations de métadonnées sont recherchées pour l'objet de modèle ComputerSystem :

```
http://example.com:9430/rest/model/meta/ComputerSystem?feed=json
```

L'exemple suivant présente la sortie JSON d'une requête de métadonnées :

```
[{"type":"java.lang.String","column":"BOOTORDER_X","length":192,"name":
"bootOrder","arrayType":false,"_class":"ObjectAttribute","timestampType
":false,"displayString":"Boot Order"}, {"type":"com.collation.platform.
model.topology.sys.zOS.ZReportFile","table":"COMPUTERSYSTILES_935A6002X
","column":"PK__ZREPORTFILES_X","length":192,"name":"ZReportfiles","arr
ayType":true,"reverseRelationship":true,"_class":"ObjectAttribute","rel
ationshipType":"com.collation.platform.model.topology.relation.AppliesT
o","timestampType":false,"displayString":"z\/OS Report File"}]
```

L'exemple suivant présente une sortie XML partielle d'une requête de métadonnées :

```
<ObjectAttribute array="22" xsi:type="coll:com.collation.platform.model
.topology.meta.ObjectAttribute">
  <name>OSRunning</name>
  <type>com.collation.platform.model.topology.sys.OperatingSystem</type>
  <arrayType>false</arrayType>
  <timestampType>false</timestampType>
  <length>192</length>
  <relationshipType>com.collation.platform.model.topology.relation.Runs
On</relationshipType>
  <reverseRelationship>true</reverseRelationship>
  <displayString>OS Running</displayString>
  <column>PK__OSRUNNING_X</column>
  <displayName />
</ObjectAttribute>
```

Service de mise à jour d'un objet de modèle :

La ressource de service de mise à jour d'un objet de modèle crée, met à jour ou supprime des objets de modèle transmis au serveur au format JSON ou XML.

Description

Utiliser le service de mise à jour d'objet de modèle pour mettre à jour ou supprimer un objet de modèle existant, ou pour ajouter un nouvel objet de modèle. Dans chaque cas, la cible de l'opération est l'objet spécifié dans le corps de la requête, au format JSON ou XML.

URL

scheme//*hostname*:*port*/rest/model/ModelObject

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

Méthodes HTTP

POST Crée ou supprime un objet de modèle, suivant la valeur du paramètre **`delete`**. L'objet à créer ou supprimer doit être spécifié dans le corps de la requête HTTP au format JSON ou XML. (Le serveur détecte automatiquement le format des données d'entrée.) Spécifiez uniquement un objet principal ; les tableaux d'objets ne sont pas pris en charge.

Si vous avez recours à cette méthode pour créer un objet, il ne faut pas qu'il existe déjà dans la base de données TADDM. (La méthode POST ne peut pas servir à mettre à jour un objet existant.)

Si vous avez recours à cette méthode pour supprimer un objet existant, seul l'identificateur global unique est requis dans les données d'entrée. Il est toutefois possible de spécifier l'objet entier. Aucune erreur n'est renvoyée si l'objet spécifié n'existe pas.

PUT Met à jour un objet existant ou en crée un nouveau. Les données du nouvel objet doivent être spécifiées dans le corps de la requête HTTP au format JSON ou XML. Spécifiez uniquement un objet principal ; les tableaux d'objets ne sont pas pris en charge.

Si l'objet spécifié existe déjà, il est mis à jour avec les nouvelles données. S'il n'existe pas, il est créé.

Si vous mettez à jour un objet existant, vous pouvez améliorer les performances en incluant uniquement l'identificateur global unique et les zones obligatoires pour la mise à jour, plutôt que l'intégralité de l'objet. Par exemple, une mise à jour de la description d'un objet OperatingSystem pourrait inclure les données suivantes :

```
[{"description":"Validated on February 4","_class":"Linux","guid":"347EE
64E4FA93139A581757EC7F3ED2D"}]
```

Tout attribut de l'objet non spécifié dans les données de mise à jour reste inchangé.

Paramètres

`delete={true|false}`
Indique si l'objet de modèle spécifié doit être supprimé. Utilisez la méthode HTTP POST et`delete=true` pour supprimer un objet.

`feed={json|xml}`

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

mssGuid=*value*
Valeur de l'identificateur global unique du système logiciel de gestion associé à l'objet. Ce paramètre est facultatif.

Résultats

Si la requête aboutit, le serveur renvoie le code retour HTTP 200.

Dans l'exemple suivant, l'objet de modèle spécifié par les données d'entrée est supprimé :

```
http://example.com:9430/rest/model/ModelObject?delete=true
```

Service de reconnaissance :

La ressource de service de reconnaissance démarre une reconnaissance avec ou sans profil.

Description

Utilisez ce type de requête pour démarrer une reconnaissance en utilisant un profil défini, ou sans profil.

URL

scheme//*hostname*:*port*/rest/discovery/start/*run_name*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

run_name
Nom de l'opération de reconnaissance.

Méthodes HTTP

POST Démarre une reconnaissance. Il ne faut pas qu'une reconnaissance soit déjà en cours. Vous devez soumettre la demande d'URL à l'aide de l'opérateur HTTP POST.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

guids=*values*
Un ou plusieurs identificateurs globaux uniques d'objets ayant déjà été préalablement reconnus. Spécifiez ce paramètre pour exécuter une nouvelle reconnaissance sur des objets existants, si vous avez activé la fonction Rediscovery (recommencer une reconnaissance).

profile=*profile_name*
Nom du profil à utiliser. Le profil spécifié doit exister.

scope=*values*
Une ou plusieurs portées, séparées par une virgule. Chaque valeur peut être :

- Un nom de portée défini
- Une adresse IP ou un nom d'hôte spécifique (par exemple, `192.168.1.71` ou `server.example.com`)
- Une adresse IP spécifique à exclure, entre parenthèses (par exemple, `(192.168.1.71)`)
- Un intervalle d'adresses IP (par exemple, `10.10.10.1-10.10.10.20`)
- Un sous-réseau (par exemple, `10.10.20.0/255.255.255.0`

Une adresse IP, un intervalle d'adresses ou un sous-réseau peut être entouré de parenthèses pour indiquer qu'il convient de l'exclure de la portée. Par exemple, `192.168.1.1-192.168.1.72,(192.168.1.71)` inclut toutes les adresses IP de l'intervalle spécifié sauf 192.168.1.71.

locationTag=*value*
Valeur de la balise d'emplacement à définir pour les objets créés au cours d'une reconnaissance.

Fix Pack 3 **addressSpace=*value***
Nom d'espace adresse défini pour tous les objets IpAddress ou IpNetwork créés lors d'une reconnaissance.

Exemple d'entrée

Cet exemple lance une reconnaissance à l'aide du profil `de reconnaissance de niveau 3` et la portée incluant les hôtes 192.168.100.101 et 102.168.100.102. Vous pouvez utiliser n'importe quel outil ou utilitaire pouvant effectuer une demande HTTP et soumettre cette demande à l'aide de l'opérateur POST.

```
http://example.com:9430/rest/discovery/start/TestRun2?profile=Level%203
%20Discovery&scope=192.168.100.101,192.168.100.102
```

Résultats

Si la requête aboutit, le serveur renvoie le code retour HTTP 200, avec le message `Discovery start submitted` (début de la reconnaissance). Si une reconnaissance est déjà en cours, la requête échoue et un message d'erreur est émis. Vous pouvez ensuite utiliser la ressource de statut de reconnaissance pour contrôler l'avancement de la reconnaissance.

Statut de reconnaissance :

La ressource de statut de reconnaissance représente le statut de reconnaissance en cours sur le serveur TADDM.

Description

Utilisez cette ressource pour vérifier l'état de l'opération de reconnaissance en cours. Les informations renvoyées sont équivalentes à celles renvoyées par la méthode Java getStatus().

URL

scheme//*hostname*:*port*/rest/discovery/status

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

Méthodes HTTP

GET Lance une requête sur le statut de la reconnaissance.

Paramètres

`feed={json|xml}`

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **`feed`**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Résultats

Le statut en cours de la reconnaissance est renvoyé dans le format spécifié. L'exemple suivant montre la réponse au format XML :

```
<status>Idle</status>
```

Exemple

Dans cet exemple, le statut de reconnaissance est vérifié :

```
http://example.com:9430/rest/discovery/status?feed=xml
```

Service de profil de reconnaissance :

La ressource du service de profil de reconnaissance affiche la liste des profils de reconnaissance définis.

Description

Servez-vous de cette ressource pour récupérer la liste de tous les profils de reconnaissance actuellement définis sur le serveur TADDM.

URL

scheme//*hostname*:*port*/rest/discovery/profiles

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

Méthodes HTTP

GET Affiche la liste des profils de la reconnaissance.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Résultats

La liste des profils de reconnaissance définis est renvoyée à l'aide du format spécifié. L'exemple suivant montre la réponse au format JSON :

```
[{"name":"profile1"},{"name":"profile2"}]
```

Exemple

Cet exemple affiche la liste des profils de la reconnaissance :

```
http://example.com:9430/rest/discovery/profiles?feed=json
```

Profil de reconnaissance :

La ressource de profil de reconnaissance représente un profil de reconnaissance défini.

Description

Servez-vous de la ressource de profil de reconnaissance pour récupérer des informations détaillées sur un profil de reconnaissance défini.

URL

scheme//*hostname*:*port*/rest/discovery/profile/*profile_name*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

profile_name
Nom d'un profil de reconnaissance défini.

Méthodes HTTP

GET Récupère les détails relatifs à un profil de reconnaissance.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Résultats

Les détails relatifs au profil de reconnaissance sont renvoyés à l'aide du format spécifié.

Exemple

Dans cet exemple, des informations sur le profil `Level 3 Discovery` sont récupérées :

```
http://example.com:9430/rest/discovery/profile/Level%203%20Discovery?feed=xml
```

Service de portée de reconnaissance :

La ressource du service de portée de reconnaissance affiche la liste des portées de reconnaissance définies.

Description

Servez-vous de cette ressource pour récupérer la liste de toutes les portées de reconnaissance actuellement définies sur le serveur TADDM.

URL

scheme//*hostname*:*port*/rest/discovery/scopes

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

Méthodes HTTP

GET Affiche la liste des portées de la reconnaissance.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Résultats

La liste des portées de reconnaissance définies est renvoyée à l'aide du format spécifié. L'exemple suivant montre la réponse au format JSON :

```
[{"name":"scope1"},{"name":"scope2"}]
```

Exemple

Cet exemple affiche la liste des portées de la reconnaissance :

```
http://example.com:9430/rest/discovery/scopes?feed=json
```

Portée de la reconnaissance :

La ressource de la portée de reconnaissance représente une portée de reconnaissance définie.

Description

Utilisez la ressource de la portée de reconnaissance pour extraire les informations détaillées sur un ensemble ou groupe de portées de reconnaissance défini.

URL

scheme//*hostname*:*port*/rest/discovery/scope/*scope_name*

où :

scheme
Modèle d'adresse URL (`HTTP:` ou `HTTPS:`).

hostname
Nom d'hôte TCP/IP ou adresse IP numérique du serveur TADDM.

port
Port TCP/IP sur le serveur TADDM pour le type de connexion que vous utilisez (9430 pour HTTP ou 9431 pour HTTPS).

scope_name
Le nom d'un ensemble ou groupe de portées de reconnaissance défini.

Méthodes HTTP

GET Récupère les détails relatifs à une portée de reconnaissance.

Paramètres

feed={json|xml}

Format à utiliser pour les données renvoyées. Spécifiez `json` ou `xml`. Ce paramètre est facultatif.

En l'absence de spécification du paramètre **feed**, le serveur utilise le format spécifié par l'en-tête HTTP Accept (`application/json` ou `application/xml`). Si cet en-tête n'est pas spécifié, les résultats sont renvoyés au format JSON.

Résultats

Les détails relatifs à la portée de la reconnaissance sont renvoyés à l'aide du format spécifié.

Exemple

Dans cet exemple, des informations sur la portée `scope1` sont récupérées :

```
http://example.com:9430/rest/discovery/scope/scope1?feed=xml
```

Interface de programme d'application de l'interface de ligne de commande

Le script `api.bat` ou `api.sh` peut être utilisé pour exécuter diverses commandes sur le serveur TADDM via l'interface de ligne de commande (Command-line interface, CLI). Par exemple, vous pouvez utiliser l'interface CLI pour démarrer une exécution de reconnaissance.

Syntaxe et paramètres de commande

Vous pouvez utiliser `api.sh` ou `api.bat` pour accéder à une partie de la fonctionnalité de l'API TADDM. Les syntaxes de la commande présentent les règles d'exécution de `api.sh` et `api.bat`.

Pour UNIX :

api.sh -u|**--user** *utilisateur* **-p**|**--password** *mdp* [**-H**|**--host** *hôte*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] *COMMAND COMMAND-PARAMETERS*

Pour Windows :

api.bat -u|**--user** *utilisateur* **-p**|**--password** *mdp* [**-H**|**--host** *hôte*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] *COMMAND COMMAND-PARAMETERS*

Paramètres communs

-u|--user *utilisateur*
L'utilisateur qui exécute la commande API.

-p|--password *mdp*
Correspond au mot de passe qui authentifie l'utilisateur.

-H|--host *hôte*
Facultatif : par défaut, le nom d'hôte du serveur TADDM est `localhost`. Si vous utilisez le paramètre **-T**, vous devez également spécifier le paramètre **-H**.

-P|--port ***port***
Facultatif : par défaut, le port du serveur TADDM est 9433.

-v|--version ***version***
Facultatif : par défaut, le nom ou le numéro de la version est 0.

-T|--truststorefile ***truststore***
Facultatif : emplacement du fichier de mémoire protégé, `jssecacerts.cert`, avec un certificat pour la connexion au serveur TADDM. Ce paramètre est obligatoire pour la connexion sécurisée à TADDM. Si vous utilisez ce paramètre, vous devez également spécifier le paramètre **-H**.

COMMAND COMMAND-PARAMETERS
Les paramètres sont différents pour chacune des commandes.

Informations supplémentaires

Pour afficher l'aide sur la commande et les paramètres de la commande, saisissez la commande suivante à partir du répertoire `$COLLATION_HOME/sdk/bin` :

Sous UNIX
`api.sh`

Sous Windows
`api.bat`

Commande changes

La commande **changes** permet d'extraire les changements d'un objet.

Syntaxe de commande

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **changes** *guid from-date* [*to-date*]

Paramètres

changes
Exécute la commande **changes**.

guid
Correspond à l'identificateur global unique de l'objet pour lequel vous souhaitez spécifier des modifications.

from-date
Correspond à la date de début de la commande **changes**. Utilisez le format `mm/dd/yy hh:mm:ss AM|PM`.

to-date
Correspond à la date de fin de la commande **changes**. Utilisez le format `mm/dd/yy hh:mm:ss AM|PM`.

Remarque : Si vous souhaitez exécuter une analyse avancée de l'historique des changements sur le serveur de reconnaissance, le paramètre `-H` doit pointer vers un serveur de stockage. Autrement, l'analyse échouera.

Exemple

Cette commande trouve toutes les modifications apportées à un objet entre les deux dates spécifiques. Entrez-la sur une ligne :

```
api.sh -u user -p password -H host -P port changes
 10A5794E86C53A0BBB10F262055CB3EA "06/06/05 12:00:00 AM" "06/08/05 12:00:00 AM"
```

Commande delete

La commande **delete** supprime les objets de la base de données TADDM.

Syntaxe de commande

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile] delete** *guid1* [*guid2 guid3 ... guidn*]

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile] delete -f**|**--file** *guid-list-file*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile] delete -m**|**--mql** *mql-query*

Paramètres

supprimer
Exécute la commande **delete**.

guid1 [guid2 guid3 ... guidn]
Correspond aux identificateurs globaux uniques des objets devant être supprimés.

-f|--file ***guid-list-file***
Correspond à l'emplacement et au nom du fichier texte qui contient la liste des identificateurs globaux uniques des objets devant être supprimés. Les identificateurs globaux uniques peuvent être séparés par un espace, une tabulation ou un caractère de retour à la ligne. Le fichier peut être généré par le script `dbquery`.

-m|--mql ***mql-query***
Correspond à la requête MQL, qui sélectionne les objets à supprimer.

Exemple

La commande suivante supprime un objet avec l'identificateur global unique spécifié. Entrez la commande sur une seule ligne.

```
api.sh -u user -p password -H host -P port delete
 10A5794E86C53A0BBB10F262055CB3EA
```

La commande suivante supprime les objets avec les identificateurs globaux uniques spécifiés. Entrez la commande sur une seule ligne.

```
api.sh -u user -p password -H host -P port delete
 C172810FD1CF3E108B8127BC47D2667B 059E4D85B34C32D1B5A80D9E2DB09EBD
35D21D3CA08539908DC1762D26897FB6
```

La commande suivante supprime les objets sélectionnés par la requête MQL spécifiée. Entrez la commande sur une seule ligne.

```
api.sh -u user -p password -H host -P port delete
 --mql "select * from AppServer where objectType == 'SAS'"
```

La commande suivante supprime les objets sur la base de la liste des identificateurs globaux uniques présente dans le fichier texte. Le script `dbquery` permet de générer `guid-list-file`. Entrez la commande sur une seule ligne.

```
dbquery.sh -u user -p password -q "select guid_c from BB_APPSERVER6_V
where objectType_C like '%SAS'" > /tmp/guidListToDelete.txt
api.sh -u user -p password -H host -P port delete -f /tmp/guidListToDelete.txt
```

Commande discover

La commande **discover** démarre ou interrompt l'exécution d'une reconnaissance.

Syntaxe de commande

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **discover start** [**--name** *run-name*] [**--profile** *profile-name*] [**--locationTag** *location-tag*] [**-a** | **-addressSpace** *addressSpace*] *scope-element1* | *scope-set1* | *scope-group1 scope-element2* | *scope-set2* | *scope-group2 ... scope-elementn* | *scope-setn* | *scope-groupn*

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **discover abort** | **status**

Paramètres

discover
Exécute la commande **discover**.

start *scope-element1 scope-element2 ... scope-elementn*
Démarre une reconnaissance avec les éléments de périmètre spécifiés. L'élément de périmètre peut être un nom de périmètre existant, ou :

- Une adresse IP spécifique : 192.168.1.71
- Une adresse IP spécifique exclue : 192.168.1.71(exclude), (192.168.1.71) ou 192.168.1.71(exc)
- Une plage ou une plage à exclure : 10.10.10.1-10.10.10.20, or (10.10.10.1-10.10.10.20)
- Un réseau (sous-réseau) ou un réseau à exclure : 10.10.20.0/255.255.255.0, ou (10.10.20.0/255.255.255.0)

start *scope-set1 scope-set2 ... scope-setn*
Démarre une reconnaissance avec les ensembles de portée spécifiés.

start *scope-group1 scope-group2 ... scope-groupn*
Démarre une reconnaissance avec le groupe de portées spécifié.

--name *run-name*
Désigne le nom de l'exécution de l'opération de reconnaissance.

--profile *profile-name*
Utilise le profil spécifié par *profile name* pour la reconnaissance.

--locationTag *location-tag*
Indique la balise d'emplacement utilisée pour cette reconnaissance.

Fix Pack 3 **-a | -addressSpace *addressSpace***
Indique le nom d'espace adresse défini pour tous les objets IpAddress ou IpNetwork créés lors de la reconnaissance démarrée avec **api.sh** ou **api.bat**.

abort|stop
Arrête une reconnaissance exécutée sur l'hôte spécifié.

status

Renvoie l'état de reconnaissance sur l'hôte spécifié, parmi les valeurs suivantes :

- En cours d'exécution
- En sommeil

Exemples

- Cette commande reconnaît le sous-réseau 10.10.10.0/24 à l'aide d'un profil de reconnaissance de niveau 1. Entrez-la sur une ligne :

```
api.sh -u user -p password -H host discover start
 --profile "Level 1 Discovery" "10.10.10.0/255.255.255.0"
```

- Cette commande reconnaît l'ensemble de portées nommé MyScope à l'aide d'un profil de reconnaissance de niveau 2. Entrez-la sur une ligne :

```
api.sh -u user -p password -H host
 discover start --profile "Level 2 Discovery" "MyScope"
```

- Cette commande reconnaît l'ensemble de portées nommé MyScope à l'aide d'un profil de reconnaissance de niveau 3 excluant un hôte 1.2.3.4 et incluant une plage 2.3.4.5-2.3.4.7. Entrez-la sur une ligne :

```
api.sh -u user -p password -H host discover start
--profile "Level 3 Discovery" "MyScope" "(1.2.3.4)" "2.3.4.5-2.3.4.7"
```

- Cette commande reconnaît l'ensemble de portées nommé MyScopeSet à l'aide d'un profil de reconnaissance de niveau 1. Entrez-la sur une ligne :

```
api.sh -u user -p password discover start
 --profile "Level 1 Discovery" "MyScopeSet"
```

- Cette commande reconnaît le groupe de portées nommé MyScopeGroup à l'aide d'un profil de reconnaissance de niveau 1. Entrez-la sur une ligne :

```
api.sh -u user -p password discover start
 --profile "Level 1 Discovery" "MyScopeGroup"
```

Commande de reconnaissance d'équilibrage de charge

Fix Pack 2

La commande **discoverloadbalanced** permet d'exécuter une reconnaissance en mode d'équilibrage de charge, ce qui implique une reconnaissance continue.

A propos de la reconnaissance à équilibrage de charge

La reconnaissance continue à équilibrage de charge est obtenue à l'aide d'un pool de serveurs de reconnaissance. Cela permet de maximiser l'utilisation des serveurs et protège contre la reprise, empêchant ainsi les reconnaissances en cours d'être interrompues.

Les environnements reconnus peuvent être séparés en zones, chacune ayant son propre pool de serveurs. En outre, chaque reconnaissance peut être exécutée en entrant des ensembles d'adresses IP pour définir la portée de la reconnaissance.

Le serveur PSS (PrimaryStorageServer) fonctionne comme un contrôleur de reconnaissance à équilibrage de charge. Il alloue la portée et les ressources à chaque serveur de reconnaissance. Si le PSS échoue, toutes les reconnaissances en cours seront achevées, à l'exception des nouvelles reconnaissances attribuées.

Le serveur PSS contrôle la reconnaissance en plaçant le travail dans une file d'attente mais ne soumet pas les tâches au serveur de reconnaissance ; au lieu de cela, chaque serveur de reconnaissance participant demande et récupère activement les tâches de la file d'attente lorsqu'il est prêt, puis indique "en cours" quand il commence la reconnaissance. Si aucun signal "en cours" n'est reçu, le serveur PSS réalloue le travail.

Lorsqu'il s'exécute en mode continu (ou d'équilibrage de charge), le serveur de reconnaissance démarre une autre reconnaissance dès que les unités d'exécution DiscoveryWorker sont disponibles pour utilisation, même si la reconnaissance

précédente est toujours en cours. La commande de reconnaissance continue est disponible uniquement pour le serveur de stockage principal (en mode entreprise) et pour le serveur de domaine (en mode domaine unique).

Conseil : L'interface utilisateur affiche les données de stockage des détecteurs comme étant en cours d'exécution. Ces données ne reflètent **pas** le nombre d'unités d'exécution qui exécutent une reconnaissance.

Scénarios de reconnaissance à équilibrage de charge

Reconnaissance à haute disponibilité avec équilibrage de charge en fonction du groupe de portées

Un utilisateur exécute une reconnaissance sur un groupe de portées défini via une interface de ligne de commande. Le pool de serveurs est utilisé pour optimiser la charge de travail de reconnaissance en fonction des portées (d'une seule à plusieurs portées) faisant partie du groupe de portées.

Génération dynamique du groupe de portées

Un utilisateur exécute une reconnaissance à équilibrage de charge à l'aide d'une liste d'adresses IP fournies sous forme d'un fichier. Cette opération créé un groupe de portées avec des portées enfant générées automatiquement, les adresses IP étant divisées entre les portées enfant.

Syntaxe de la commande

Dans le cadre d'une reconnaissance standard lancée à l'aide de la commande **`discover`**, la fonctionnalité de reconnaissance d'équilibrage de charge continue (**`discoverloadbalanced`**) est contrôlée à l'aide de la commande **`api.sh`**.

Paramètres

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced start --poolName <nom_pool> --scopeGroup <groupe_portée> [--profile <nom_profil>]

Démarre la reconnaissance d'équilibrage de charge continue sur chaque portée de `<groupe_portée>` et sur chaque serveur appartenant au pool de reconnaissance spécifié dans `<nom_pool>`

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced startFromFiles --files <chemin_fichier_1,chemin_fichier_2,...> --maxScopeSize <taille> [--profile <nom_profil>]

Démarre la reconnaissance d'équilibrage de charge continue sur chaque portée passée via l'argument `--files`. Chaque fichier est analysé, et un nouveau groupe nommé d'après le nom du fichier (sans extension) est créé. Dans chaque groupe, des portées sont ajoutées avec la taille maximale spécifiée dans `maxScopeSize`. La valeur de `poolName` est supposée être la même que celle de `groupName`

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced startFromDirectory --dir <chemin_répertoire> --maxScopeSize <taille> [--profile <nom_profil>]

Même chose que pour l'exemple `--files`, mais démarre la reconnaissance sur chaque fichier du répertoire.

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced status

Imprime le statut actuel de la reconnaissance d'équilibrage de charge.

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced abort <nom_pool>
Abandonne la reconnaissance pour le `nom de pool` spécifié.

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced pause <nom_pool>
Arrête la reconnaissance pour le `nom de pool` spécifié ; toutes les portées de la reconnaissance exécutée repassent à l'état 'forTake'. Les reconnaissances en cours sont abandonnées.

api.sh -u <utilisateur> -p <mot_de_passe> discoverloadbalanced resume <nom_pool>
Reprend la reconnaissance suspendue pour le `nom de pool` spécifié ; toutes les portées à l'état 'forTake' sont alors disponibles pour le traitement.

Propriétés du serveur de stockage principal

Définissez les propriétés suivantes pour le serveur (ou le domaine) de stockage principal :

com.ibm.cdb.internalscheduling.discoverypool.scopePrefix
Indique le préfixe des portées créées automatiquement (pour les options startFromFiles et startFromDirectory)

Valeur par défaut :
`com.ibm.cdb.internalscheduling.discoverypool.scopePrefix=_auto_`

com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive
Indique la durée, en secondes, pendant laquelle la reconnaissance est supposée s'exécuter sans qu'aucune information relative au statut ne soit reçue du serveur de reconnaissance.

Une fois ce temps écoulé, la portée repasse à l'état 'forTake'. Elle est alors disponible au serveur qui répond encore.

Valeur par défaut :
`com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive=180`

Propriétés du serveur de reconnaissance

Définissez les propriétés suivantes pour le serveur (ou le domaine) de reconnaissance :

com.ibm.cdb.internalscheduling.discoverypool.enabled
Configure le serveur de reconnaissance pour qu'il fasse partie du pool de reconnaissance.

Valeur par défaut :
`com.ibm.cdb.internalscheduling.discoverypool.enabled=false`

com.ibm.cdb.internalscheduling.discoverypool.name
Définit le nom de pool d'un serveur de reconnaissance.

Valeur par défaut :
`com.ibm.cdb.internalscheduling.discoverypool.name=DEFAULT`

com.ibm.cdb.internalscheduling.discoverypool.checkinginterval
Indique le temps, en secondes, pendant lequel le serveur de reconnaissance vérifie les nouveaux travaux devant être exécutés (s'ils sont prêts), et la fréquence à laquelle PSS doit être informé d'un travail en cours.

L'intervalle spécifié doit être inférieur à celui de la propriété `com.ibm.cdb.internalscheduling.discoverypool.timeToNonAlive` définie pour le serveur de stockage principal. Sinon, les portées peuvent être renvoyées en vue d'une nouvelle exécution.

Valeur par défaut :
`com.ibm.cdb.internalscheduling.discoverypool.checkinginterval=20`

Remarque : Les journaux associés à la reconnaissance continue sont stockés dans ApiServer.log sur le domaine, ou dans le serveur de stockage principal.

Commande export

La commande **export** permet d'exporter des données pour les objets de modèle de niveau supérieur dans la base de données TADDM.

Syntaxe de commande

api.sh | **api.bat** **-u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **export** [**--mssguid** *mss-guid* | **--mssname** *mss-name*] [**--maxfilesize** *size*] *local-directory-to-write-data*

Paramètres

export
Exécute la commande **export**.

--mssguid ***mss-guid*** | **--mssname** ***mss-name***
Correspond à l'identificateur global unique ou au nom du système logiciel de gestion. Seules les données associées au système logiciel de gestion spécifié sont exportées.

Le nom du système logiciel de gestion est disponible dans l'interface utilisateur. Dans la sous-fenêtre Détails d'un objet de modèle, ouvrez l'onglet **Informations MSS**. Le **Nom d'instance du sous-composant** correspond au nom du système logiciel de gestion, par exemple `LinuxComputerSystemSensor`. Si vous souhaitez exporter tous les objets reconnus par ce détecteur, exécutez la commande suivante :

```
export --mssName LinuxComputerSystemSensor
```

Pour obtenir le nom et l'identificateur global unique du système logiciel de gestion, exécutez la commande suivante :

```
api.sh -u user -p password find -d 1 ManagementSoftwareSystem
```

Cette commande renvoie une liste d'objets de modèle associés à un ensemble d'informations, par exemple :

```
<ManagementSoftwareSystem array="1"
    guid="MY_GUID" xsi:type="coll:com.collation.platform.model.topology.process.
ManagementSoftwareSystem">
    <manufacturerName>IBM</manufacturerName>
    <productName>TADDM</productName>
    <hostname>hostname.domain</hostname>
    <subcomponent>Discovery</subcomponent>
    <subcomponentInstanceName>IvmSensor</subcomponentInstanceName>
    <displayName>IBM:TADDM:hostname.domain:Discovery:IvmSensor</displayName>
    <bidiFlag>3</bidiFlag>
</ManagementSoftwareSystem>
```

Dans cet exemple, l'identificateur global unique du système logiciel de gestion est `MY_GUID` et son nom est `IvmSensor`. Si vous souhaitez exécuter la commande **export** avec ces données, utilisez l'une des commandes suivantes :

```
export --mssguid MY_GUID
```

ou

```
export --mssname IvmSensor
```

Remarque : Si vous exécutez la commande comprenant l'identificateur global unique, utilisez la valeur appropriée à partir de la requête API.

--maxfilesize ***size***
Correspond à la taille maximale des fichiers exportés, en octets.

local-directory-to-write-data
Désigne le nom du répertoire vers lequel les données sont exportées.

Exemple

Cette commande exporte les objets de modèle de niveau supérieur vers le répertoire spécifique :

```
api.sh -u user -p password -H host export directory/
```

Commande find

La commande **find** recherche un ensemble d'objets et renvoie une représentation XML.

Syntaxe de commande

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **find** [**--depth** *depth*] [**--indent** *num-spaces*] [**-o**|**--outfile** *local-file-to-write-to* [**-x --maxfilesize** *size*]] [**-s --suppress** *list-of-classes-to-suppress*] *root*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **find** [**--depth** *depth*] [**--indent** *num-spaces*] **--guid** *object-guid*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **find** [**--depth** *depth*] [**--indent** *num-spaces*] [**--changetype** *type* **--from** *from-date* [**--end** *end-date*]] *root*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **find** [**--depth** *depth*] [**--indent** *num-spaces*] [**--mssguid** *mss-guid*|**--mssname** *mss-name*] *mql-query*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **find --count** *mql-query*

Paramètres

find
Exécute la commande **find**.

--depth ***depth***
Indique le niveau de l'arborescence de résultat à construire.

L'interrogation d'une grande quantité de données ou la spécification de plusieurs niveaux peut entraîner des messages signalant un manque de

mémoire. Pour éviter les problèmes de mémoire, limitez la valeur de profondeur ou augmentez la taille de segment maximum de la mémoire de la machine virtuelle Java. Si possible, n'utilisez pas une valeur de profondeur supérieure à 3. Vous pouvez augmenter la mémoire avec l'option JVM `-Xmx` dans `api.bat` ou `api.sh`.

--indent *num spaces*
Désigne l'indentation à utiliser pour la sortie XML obtenue.

--changetype *type*

Le type de modification provient-il de l'une des valeurs suivantes :

0 Création

1 Mis à jour

2 Supprimé

3 Créations et mises à jour

4 Tous les changements

--from *from-date*
Correspond à la date de début du paramètre de modification. Utilisez le format `mm/dd/yy hh:mm:ss AM|PM`.

--end *end-date*
Correspond à la date de fin du paramètre de modification. Utilisez le format `mm/dd/yy hh:mm:ss AM|PM`.

-o|--outfile *local-file-to-write-to*
Correspond au nom du fichier vers lequel rediriger la sortie de la commande **find**.

-x |--maxfilesize *size*
Correspond au fichier de sortie qui peut être décomposé en plusieurs petits fichiers, dont vous spécifiez la taille maximale en octets. La sortie est divisée en plusieurs fichiers sous la taille de fichiers maximum, dès que cela est possible.

-s|--suppress *list-of-classes-to-suppress*
Correspond à une liste des classes à omettre à partir des résultats de la recherche. Il s'agit de classes de nom d'objet de modèle, comme ComputerSystem ou OperatingSystem.

--guid *object-guid*
Correspond à l'identificateur global unique de l'objet pour lequel la commande **find** est en cours d'exécution.

--mssguid *mss-guid*|--mssname *mss-name*
Correspond à l'identificateur global unique ou au nom du système logiciel de gestion.

--count *mql-query*
Renvoie le nombre d'objets répondant à la requête MQL.

mql-query
Correspond à la requête spécifiée à l'aide de MQL (Model Query Language), par exemple, SELECT attributes FROM object type [WHERE expression]. Dans cet argument, vous pouvez utiliser des noms abrégés ou longs pour les types d'objet. Pour plus d'informations sur les noms de classe et les demandes MQL, voir les concepts associés.

root
Indique si l'objet de modèle va servir de racine pour la sortie XML obtenue. Dans cet argument, vous pouvez utiliser des noms abrégés ou longs pour les types d'objet. Pour plus d'informations sur les noms de classe, voir le concept associé.

Exemples

- Cette commande détecte les systèmes informatiques et enregistre les résultats dans le fichier `cs_output.xml`, dont la taille maximale est de 1 000 octets :

  ```
  api.sh -u user -p password -H host -P port find -o cs_output.xml -x 1000 ComputerSystem
  ```

- Cette commande compte le nombre d'objets ComputerSystem dans la base de données :

  ```
  api.sh -u user -p password find --count "select * from ComputerSystem"
  ```

- Cette commande limite la recherche à un niveau de profondeur prédéfini. Les résultats sont sauvegardés dans le fichier `cs_output.xml` :

  ```
  api.sh -u user -p password -H host -P port find --depth depth -o cs_output.xml ComputerSystem
  ```

Commande import

La commande **import** permet d'importer les données dans la base de données TADDM.

Syntaxe de commande

api.sh | **api.bat** **-u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **import** [**--timeout** *time*] [**--mssguid** *mss-guid* **--mssname** *mss-name*] [**--maxfilesize** *size*] *local-directory-to-read-data-from*

Paramètres

import
Exécute la commande **import**.

--timeout ***time***
Correspond à la valeur du délai d'attente, utile pour les importations de fichiers très volumineux. Spécifiez la valeur en secondes.

--mssguid ***mss-guid***|**--mssname** ***mss-name***
Correspond à l'identificateur global unique ou au nom du système logiciel de gestion auquel les données importées sont associées.

local-directory-to-read-data-from
Désigne le nom du répertoire vers lequel les données sont importées.

Exemple

Cette commande importe les données vers TADDM. La commande tente d'importer tous les fichiers dans le répertoire spécifié. Si la commande rencontre un fichier XML incorrect, une exception est renvoyée mais la commande continue l'importation.

```
api.sh -u user -p password -H host import directory/
```

Commande merge

La commande **merge** fusionne des éléments de configuration en fonction de leurs identificateurs globaux uniques.

Syntaxe de la commande

api.sh | api.bat -u | --user *user* **-p | --password** *password* [**-H | --host** *host*] [**-P | --port** *port*] [**-T | --truststorefile**] **merge** *DurableCI_GUID TransientCI_GUID* [**type** *type*]

Paramètres

merge
Exécute la commande **merge**.

DurableCI_GUID
Identificateur global unique qui doit être conservé après la fusion avec l'autre identificateur global unique.

TransientCI_GUID
Identificateur global unique qui doit être fusionné dans l'identificateur global unique durable et supprimé de la base de données.

type ***type***
Type du processus de fusion. Les deux valeurs possibles sont les suivantes :

- *0* : correspond au processus de fusion superficiel, ce qui signifie que seuls les objets principaux sont fusionnés. Les objets enfant de l'identificateur global unique temporaire sont remplacés par les objets enfant de l'identificateur global unique durable. Le processus de fusion superficiel est le type de fusion par défaut.
- *1* : correspond au processus de fusion approfondi, ce qui signifie que les objets principaux et leurs objets enfant sont fusionnés.

Remarque : Actuellement, seul le type de processus de fusion superficiel est pris en charge. Lorsque vous indiquez la valeur *1* pour le paramètre **type**, le type superficiel est utilisé. Le type approfondi sera introduit ultérieurement.

Important : Lorsque vous fusionnez des éléments de configuration, ceux-ci doivent être de la même classe. Par exemple, vous pouvez fusionner deux objets ComputerSystem, mais vous ne pouvez pas fusionner un objet ComputerSystem avec un objet ApplicationServer.

Exemple

La commande suivante effectue une fusion superficielle des deux identificateurs globaux uniques indiqués. Entrez la commande sur une seule ligne.

```
api.sh -u user -p password -H host -P port merge 10A5794E86C53A0BBB10F262055CB3EA
C172810FD1CF3E108B8127BC47D2667B type 0
```

Commande naming

La commande **naming** renvoie les identificateurs globaux uniques associés à un élément de configuration (EC). La commande renvoie uniquement les identificateurs globaux uniques des éléments de configuration de niveau supérieur du fichier XML.

Syntaxe de commande

api.sh | api.bat -u | --user *user* **-p | --password** *password* [**-H | --host** *host*] [**-P | --port** *port*] [**-T | --truststorefile**] **naming -f** *model-object-xml-file*

Paramètres

naming
Exécute la commande **naming**.

-f model-object-xml-file
L'emplacement et le nom du fichier XML contenant l'élément de configuration (objet de modèle).

Exemple

Cette commande affiche les identificateurs globaux uniques des éléments de configuration du fichier XML :

```
api.sh -u user -p password -H host naming sample.xml
```

Commande rediscover

La commande **rediscover** reconnaît à nouveau des objets.

Remarque : Avant d'exécuter la nouvelle reconnaissance, vérifiez que le paramètre **com.collation.rediscoveryEnabled** est défini sur `true`. Les objets que vous souhaitez reconnaître à nouveau doivent avoir été reconnus au moins une fois auparavant.

Syntaxe de commande

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* **rediscover** *guid1* [*guid2 guid3 ... guidn*]

Paramètres

rediscover
Exécute la commande **rediscover**.

guid1 [guid2 guid3 ... guidn]
Correspondent aux identificateurs globaux uniques des objets devant être à nouveau reconnus.

Exemple

La commande suivante reconnaît à nouveau les objets avec les identificateurs globaux uniques spécifiés. Entrez la commande sur une seule ligne.

```
./api.sh -u administrator -p collation rediscover 4C778F231DB03FCF815E38EAD7CB1D66
B609D10039C23AE9A51E433EC311A9EE F503F3B70DF93587A635D574A78B248A
```

Commande servers

La commande **servers** affiche des informations sur les serveurs dans un déploiement de serveurs en continu.

Syntaxe de commande

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile] servers getservers**

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile] servers getdiscoveryservers**

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile] servers getdiscoveryserverstatus**

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile] servers getstorageservers**

api.sh | api.bat -u | --user *user* **-p | --password** *password* **[-H | --host** *host*] **[-P | --port** *port*] **[-T | --truststorefile] servers getstorageserverstatus**

api.sh | api.bat -u | --user *user* **-p | --password** *password* **[-H | --host** *host*] **[-P | --port** *port*] **[-T | --truststorefile] servers getlocalserver**

api.sh | api.bat -u | --user *user* **-p | --password** *password* **[-H | --host** *host*] **[-P | --port** *port*] **[-T | --truststorefile] servers getlocalserverstatus**

Paramètres

servers
Exécute la commande **servers**.

getservers
Répertorie tous les serveurs de stockage et de reconnaissance en cours d'exécution.

getdiscoveryservers
Répertorie tous les serveurs de reconnaissance en cours d'exécution.

getdiscoveryserverstatus
Affiche le statut détaillé et les informations de performance pour tous les serveurs de reconnaissance.

getstorageservers
Répertorie tous les serveurs de stockage en cours d'exécution.

getstorageserverstatus
Affiche le statut détaillé et les informations de performance pour tous les serveurs de reconnaissance.

getlocalserver
Affiche les informations relatives au serveur local.

getlocalserverstatus
Affiche le statut détaillé et les informations de performance pour le serveur local.

Exemples

- Cette commande répertorie tous les serveurs de reconnaissance et de stockage en cours d'exécution :

  ```
  api.sh -u user -p password -H host -P port servers getservers
  ```

- Cette commande répertorie tous les serveurs de reconnaissance en cours d'exécution :

  ```
  api.sh -u user -p password -H host -P port servers getdiscoveryservers
  ```

- Cette commande montre le statut détaillé et des informations sur les performances de tous les serveurs de stockage :

  ```
  api.sh -u user -p password -H host -P port servers getstorageserverstatus
  ```

Commande sync

La commande **sync** démarre la synchronisation du serveur de domaine

Syntaxe de commande

api.sh | api.bat -u | --user *user* **-p | --password** *password* **[-H | --host** *host*] **[-P | --port** *port*] **[-T | --truststorefile] sync start** *domain*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **sync status** *domain*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **sync logs** *domain*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **sync stop** *domain*

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **sync delete** *domain*

Paramètres

sync
Exécute la commande **sync**.

début
Démarre la synchronisation du serveur de domaine.

status
Affiche l'état d'une synchronisation de serveur de domaine.

logs
Affiche les fichiers journaux de synchronisation du serveur de domaine.

stop
Arrête la synchronisation d'un serveur de domaine.

delete
Supprime la synchronisation d'un serveur de domaine.

domain
Correspond au nom de domaine ajouté au serveur de synchronisation et non au nom d'hôte du serveur du domaine.

Exemple

Cette commande démarre la synchronisation du serveur de domaine :

```
api.sh -u user -p password -H host sync domain
```

Commande topology

La commande **topology** affiche l'état des groupes de topologie.

Syntaxe de commande

api.sh|**api.bat -u**|**--user** *user* **-p**|**--password** *password* [**-H**|**--host** *host*] [**-P**|**--port** *port*] [**-T**|**--truststorefile**] **topology groups**

Paramètres

topology
Exécute la commande **topology**.

groups
Affiche l'état détaillé des groupes de topologie.

Exemple

Cette commande affiche l'état détaillé des groupes de topologie :

```
api.sh -u user -p password  topology groups
```

Commande version

La commande **version** gère les versions dans IBM Tivoli Application Dependency Discovery Manager (TADDM).

Syntaxe de commande

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **version** [**-c** | **--create** *version-name version-description*]

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **version** [**-e** | **--createempty** *version-name version-description*]

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **version** [**-d** | **--delete** *version-id-or-name*]

api.sh | **api.bat -u** | **--user** *user* **-p** | **--password** *password* [**-H** | **--host** *host*] [**-P** | **--port** *port*] [**-T** | **--truststorefile**] **version getall**

Paramètres

version
Exécute la commande **version**.

-c|--create ***version-name version-description***
Crée une version avec le nom fourni.

-e|--createempty ***version-name version-description***
Crée une version vide avec le nom fourni.

-d|--delete ***version-id-or-name***
Supprime la version spécifiée.

getall
Affiche toutes les versions existantes.

Exemples

- Cette commande crée une version :

  ```
  api.sh -u user -p password -H host -P port version -create "version1.0"
   "Il s'agit de la version d'origine"
  ```

- Cette commande supprime une version :

  ```
  api.sh -u user -p password -H host -P port version -delete "version1.0"
  ```

Développement d'extensions de serveur personnalisé

Vous pouvez utiliser des extensions de serveur personnalisé pour reconnaître les cibles pour lesquelles TADDM ne propose aucun support intégré spécifique ou un support limité ou pour ajouter une fonctionnalité à TADDM.

Les extensions de serveur personnalisé fournissent une interface de programme d'application (Application Programming Interface, API) qui vous permet de créer des programmes qui définissent des attributs définis dans le modèle de données communes (Common Data Model, CDM) ou des attributs étendus que vous avez ajoutés dans le CDM à l'aide du portail de gestion de données.

Ces extensions basées sur Jython s'exécutent dans le moteur de reconnaissance TADDM qui fournit aux extensions de serveur personnalisé une infrastructure permettant de contrôler de nombreux blocs de construction de détecteur dans TADDM.

Les extensions de serveur personnalisé proposent les fonctions suivantes :

- Vous pouvez utiliser l'interface utilisateur TADDM et l'interface de programme d'application pour visualiser les attributs reconnus.
- Les messages d'extension de serveur personnalisé sont consignés dans le journal Discovery Manager et dans le journal du détecteur de l'ordinateur approprié (ou dans les journaux CustomAppServerSensor si la division des journaux de détecteur est activée).
- Aucun logiciel supplémentaire n'est requis pour utiliser le système.

Limitation : Vous ne pouvez pas utiliser de détecteurs basés sur un script pour créer des extensions de serveur personnalisées.

Limitation : Lorsque vous étendez la reconnaissance sur les systèmes d'exploitation Windows pour exécuter des commandes qui renvoient la sortie avec des caractères Unicode, ces caractères ne sont pas enregistrés.

Fix Pack 3 Dans TADDM version 7.3.0.3 et ultérieures, l'enregistrement des caractères Unicode est pris en charge. Cependant, vous devez commencer par envoyer la sortie dans un fichier avant de lire ce fichier. L'exemple suivant affiche des commandes que vous pouvez utiliser pour effectuer ces deux opérations. Dans l'exemple, la sortie de la commande **unicodetest.bat** est envoyée dans le fichier `c:\\r.txt`.

```
os_handle.executeCommand("c:\\unicodetest.bat | out-file c:\\r.txt")
output =  os_handle.executeCommand("cmd.exe /u /c type c:\\r.txt")
```

Présentation

Vous pouvez utiliser le portail de gestion de données et les interfaces de programme d'application des extensions de serveur personnalisé pour définir des attributs intégrés ou étendus.

Le développement d'une extension de serveur personnalisé implique d'identifier les attributs intégrés et étendus que vous voulez définir et d'ajouter les attributs étendus au Common Data Model (CDM). Une fois ces opérations effectuées, vous devez écrire l'application pour définir les attributs et vérifier que la collecte des attributs s'effectue comme prévu.

La procédure de développement d'une extension de serveur personnalisé est présentée ci-après :

1. Identifiez les attributs intégrés ou étendus que vous voulez collecter.
2. Si vous avez identifié des attributs étendus, ajoutez-les au Common Data Model à l'aide du portail de gestion de données.

   Pour plus d'informations, voir «Gestion des attributs étendus», à la page 136.
3. Développez l'application pour définir les attributs à l'aide de l'interface de programme d'application des extensions de serveur personnalisé.

   Pour plus d'informations sur l'interface de programme d'application des extensions de serveur personnalisé, voir «API des extensions de serveur personnalisé», à la page 136. Pour un exemple d'application, voir «Exemple d'application d'extension de serveur personnalisé», à la page 166.
4. Lancez l'application d'extension de serveur personnalisé.

5. A l'aide du portail de gestion de données, vérifiez que les attributs sont définis comme prévu.

Gestion des attributs étendus

Vous devez définir les attributs étendus pour pouvoir collecter les attributs à l'aide de votre application de serveur personnalisé.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le portail de gestion de données afin d'ajouter ou de supprimer des attributs étendus pour un type de composant dans le modèle de données communes.

Le tableau 30 décrit les paramètres que vous pouvez spécifier pour les attributs étendus.

Tableau 30. Attributs étendus

Zone	Description
Type de composant	Identifie le type de composant.
Nom d'attribut étendu	Désigne le nom de l'attribut étendu.
Type d'attribut étendu	Indique le type de l'attribut étendu.
Nom d'attribut hérité	Si cette classe est une sous-classe d'une autre classe dans le Common Data Model, et si les attributs étendus ont été définis pour la classe parent, ces attributs sont répertoriés ici.
Type d'attribut hérité	Désigne le type de l'attribut hérité.

Procédure

Pour spécifier des attributs étendus, procédez comme suit :

1. Lancez le portail de gestion de données.
2. Dans le menu principal, choisissez **Modifier** > **Attributs étendus**.
3. Dans la liste **Type de composant**, sélectionnez celui de votre choix. La fenêtre Définir des attributs étendus affiche les attributs étendus définis actuellement pour le type de composant sélectionné.
4. Pour ajouter un attribut étendu, cliquez sur **Nouveau**. Entrez le nom de l'attribut et le type d'attribut dans les zones correspondantes et cliquez sur **OK**. Le système ajoute le nom d'attribut à la liste des attributs étendus.
5. Pour supprimer un attribut étendu, procédez comme suit :
   a. Dans la liste **Type de composant**, sélectionnez celui de votre choix.
   b. Sélectionnez l'attribut que vous voulez supprimer dans la fenêtre Définir des attributs étendus.
   c. Cliquez sur **Supprimer**.
6. Cliquez sur **OK** pour enregistrer les modifications et fermer la fenêtre, ou cliquez sur **Annuler** pour fermer la fenêtre sans les enregistrer.

API des extensions de serveur personnalisé

L'interface de programme d'application des extensions de serveur personnalisé fournit un ensemble de fonctions qui vous permettent de créer des applications Jython susceptibles de récupérer des informations sur l'exécution des processus,

d'exécuter des commandes, de capturer des fichiers, de normaliser des données, de créer de nouveaux objets CDM et de définir des attributs ou des attributs étendus dans le Common Data Model.

Cette section décrit les actions que vous devez préalablement effectuer avant de commencer à utiliser l'interface de programme d'application des extensions de serveur personnalisé, puis présente les types de fonction disponibles dans cette interface. Elle contient également une description de chaque classe de fonction disponible, avec un exemple de code indiquant les éléments les plus courants à inclure dans vos applications.

Vous pouvez également étendre les modèles de serveur et de système informatique personnalisés en exécutant des scripts Jython. Pour plus d'informations, voir la rubrique *Extension de modèles de serveur et de système informatique personnalisés* dans le *Guide d'utilisation* de TADDM.

Prérequis pour utiliser l'API des extensions de serveur personnalisé

Vous devez connaître les concepts clés avant d'utiliser l'API des extensions de serveur personnalisé.

Avant de créer des applications à l'aide de l'API des extensions de serveur personnalisé, vous devez comprendre les concepts suivants :

- Common Data Model (CDM)

  Si vous utilisez les extensions de serveur personnalisé pour créer de nouveaux ModelObjects CDM, vous devez définir au moins une règle de dénomination, sinon TADDM ne pourra pas générer d'identificateur global unique pour l'objet et le détecteur dans lequel l'extension est exécutée échouera (en émettant une erreur de stockage).
- Comment faire pour que l'application configure l'environnement Jython à utiliser avec TADDM et l'API des extensions de serveur personnalisé.

  Vous pouvez utiliser le fichier `sensorstub.py` dans le répertoire `$COLLATION_HOME/lib/sensor-tools` comme base pour votre application d'extension de serveur personnalisé. Ce code fonctionne comme un module de remplacement qui configure correctement l'environnement Jython mais n'effectue aucune opération au niveau du système.
- Comment créer et gérer des serveurs personnalisés.

  Pour plus d'informations, voir le *Guide d'utilisation* de TADDM.

Présentation des fonctions

L'interface de programme d'application des extensions de serveur personnalisé propose plusieurs classes de fonctions destinées à vous aider à écrire des extensions de serveur personnalisé.

L'interface de programme d'application des extensions de serveur personnalisé est composée d'un ensemble de fonctions Python que vous pouvez utiliser pour exécuter des commandes et gérer des processus, effectuer des recherches DNS et avoir accès aux répertoires et fichiers sur des cibles distantes. Elle fournit également des fonctions permettant de manipuler le contrôle d'accès obligatoire et les adresses IP, et d'utiliser des descripteurs de système d'exploitation pour récupérer des informations.

En outre, l'interface de programme d'application contient un ensemble d'utilitaires vous permettant d'effectuer des tâches utiles dans vos applications.

Les catégories suivantes de fonctions sont disponibles dans l'API des extensions de serveur personnalisé.

Fonction
Utilisation de fonctionnalités telles que ExecuteCapability, MibQueryCapability ou OsInfoCapability.

Commande et processus
Exécution de commandes sur une cible, avec gestion et affichage des informations associées à un processus.

DNS et domaines
Exécution de recherches de nom de domaine et validation de noms de domaine complets.

Accès à un fichier
Liste du contenu d'un répertoire et capture de fichiers à partir de cibles distantes.

Adresse IP et MAC
Manipulation et conversion d'adresses IP et MAC.

Système d'exploitation
Création et utilisation de descripteurs de système d'exploitation pour récupérer des informations.

Chemin
Conversion des caractères de séparation de chemin Windows et Unix.

Utilitaire
Initialisation de l'interface de programme d'application de serveur personnalisé et exécution de diverses tâches utiles.

Informations de version
Détermination des numéros de version de l'interface de programme d'application.

Fonctions de fonctionnalités :

Les fonctions de fonctionnalités vous permettent d'utiliser des fonctionnalités telles que ExecuteCapability, MibQueryCapability ou OsInfoCapability. L'utilisation de ces fonctions facilite l'exécution d'une opération requise sur une cible spécifiée.

Vous pouvez utiliser une fonction de fonctionnalités pour retrouver l'usine responsable de la création de fonctionnalités pour une cible spécifiée. Le tableau 31 décrit les fonctions que vous pouvez utiliser.

Tableau 31. Fonctions de fonctionnalités

Fonction	Description
getSimpleCapabilitiesFactory	Renvoie un objet SimpleCapabilitiesFactory pour une adresse IP donnée.

Fonctions de commande et de processus :

Les fonctions de commande et de processus permettent de lancer des commandes sur une cible, ainsi que de gérer et afficher les informations associées à un processus.

Vous pouvez utiliser les fonctions de commande et processus pour lancer une commande sur une cible, en spécifiant en option une valeur de délai qui détermine

la durée d'exécution autorisée pour la commande. Vous pouvez également utiliser les fonctions pour ajouter un processus d'exécution dans le pool des processus et renvoyer la mappe de connexion, la liste des ports, la mappe du processus d'exécution et les processus serveur associés aux identificateurs du processus.

Le tableau 32 décrit les fonctions que vous pouvez utiliser.

Tableau 32. Fonctions de commande et de processus

Fonction	Description
addProcessToPool	Ajoute un processus d'exécution à un pool de processus.
executeCommand	Exécute une commande sur la cible.
executeCommandWithTimeout	Exécute une commande sur la cible avec un délai d'attente spécifiant la durée d'exécution autorisée pour la commande.
getPidConnectionMap	Renvoie un dictionnaire Python de listes Python contenant les informations suivantes : • clés : ID de processus • listes: connexions TCP des ID de processus
getPidPortList	Renvoie un dictionnaire Python de listes Python contenant les informations suivantes : • clés : ID de processus • listes : ports utilisés par le processus pour l'écoute ou la connexion
getPidToRuntimeProcessMap	Renvoie un dictionnaire Python contenant les informations suivantes : • ID de processus • informations sur le processus d'exécution
getProcessByPid	Renvoie l'objet RuntimeProcess du CDM associé à l'ID de processus donné.
getServerProcesses	Renvoie un dictionnaire Python de listes Python contenant les informations suivantes : • clés : ID de processus • listes : adresses de liaison des ports d'écoute

Fonctions du Common Data Model :

Les fonctions du Common Data Model (CDM) permettent de gérer le Common Data Model.

Vous pouvez utiliser les fonctions CDM pour créer et cloner les nouveaux objets CDM et définir la valeur des attributs étendus dans des objets CDM.

Le tableau 33, à la page 140 décrit les fonctions que vous pouvez utiliser.

Tableau 33. Fonctions du Common Data Model

Fonction	Description
cloneModelObject	Permet de créer une copie d'un ModelObject du Common Data Model.
newModelObject	Permet de créer un objet CDM.
setExtendedAttributes	Définit les valeurs des attributs étendus.

Fonctions DNS et de domaine :

Les fonctions DNS et de domaine vous permettent d'effectuer des recherches de nom de domaine et de valider les noms de domaine complets.

Vous pouvez utiliser les fonctions DNS et de domaine pour effectuer des recherches de nom pour le serveur TADDM et les noms extraits d'une configuration distante. Vous pouvez également utiliser ces fonctions pour valider un nom de domaine complet.

Le tableau 34 décrit les fonctions que vous pouvez utiliser.

Tableau 34. Fonctions DNS

Fonction	Description
getLocalDNSLookup	Effectuez une recherche de nom d'hôte sur le serveur TADDM.
getRemoteDNSLookup	Permet de rechercher un nom extrait d'une configuration distante qui risque de ne pas se résoudre sur le serveur TADDM.
validateFqdn	Vérifie un nom de domaine complet pour garantir sa conformité aux règles décrites dans la RFC 1035.

Fonctions d'accès au fichier :

Les fonctions d'accès au fichier vous permettent d'afficher la liste du contenu d'un répertoire et de capturer de fichiers à partir de cibles distantes.

Vous pouvez utiliser les fonctions d'accès au fichier pour répertorier le contenu d'un répertoire et pour capturer le contenu et les métadonnées de fichiers sur des cibles distantes. Le tableau 35 décrit les fonctions que vous pouvez utiliser.

Tableau 35. Fonctions d'accès au fichier

Fonction	Description
getFile	Permet de capturer un fichier à partir d'une cible distante, et de renvoyer les métadonnées et le contenu du fichier.
getFileWithLengthLimit	Permet de capturer un fichier, jusqu'à la longueur maximale spécifiée, à partir d'une cible distante, et de renvoyer les métadonnées et le contenu du fichier.
listDirectory	Permet de renvoyer une liste Python présentant le contenu d'un répertoire sur une cible distante.

Fonctions d'adresses IP et MAC :

Les fonctions d'adresses IP et MAC vous permettent de manipuler et de convertir des adresses IP et MAC.

Vous pouvez utiliser les fonctions d'adresses IP et MAC pour manipuler des adresses MAC, valider des adresses IP et convertir des adresses IP entre différentes représentations. Le tableau 36 décrit les fonctions que vous pouvez utiliser.

Tableau 36. Fonctions d'adresses IP et MAC

Fonction	Description
binToDot	Permet de convertir une représentation binaire d'une adresse IP en notation décimale.
bitsMaskToDottedDecimalMask	Permet de convertir la notation de masque de contrôle des données du réseau en notation décimale avec point (par exemple, 24 > 255.255.255.0).
calcNetworkAddress	Permet de calculer l'adresse réseau, à partir d'une adresse IP et d'un masque de réseau.
canonicalMac	Permet de supprimer les séparateurs ou la notation de base d'une adresse MAC pour la renvoyer sous forme d'adresse MAC hexadécimale en tant que chaîne avec caractères alphanumériques en majuscules.
classlessNotation	Permet de calculer la notation sans classe d'un réseau IP.
dotToBin	Permet de convertir une adresse IPv4 d'une notation décimale en format binaire.
ipInSubnet	Permet de déterminer si une adresse IP est un membre d'un sous-réseau donné et non l'adresse de diffusion.
networkToList	Permet de renvoyer toutes les adresses IP qui sont membres de la représentation CDM du paramètre IpNetwork.
validateIp	Permet de valider une adresse IP en notation décimale.

Fonctions de système d'exploitation :

Les fonctions de système d'exploitation vous permettent de créer et d'utiliser des descripteurs de système d'exploitation pour extraire des informations.

Vous pouvez utiliser les fonctions de système d'exploitation pour créer des descripteurs de système d'exploitation et récupérer des informations à l'aide de ces descripteurs. Le tableau 37, à la page 142 décrit les fonctions que vous pouvez utiliser.

Tableau 37. Fonctions de système d'exploitation

Fonction	Description
getAppTarget	Renvoie les informations suivantes : • descripteur du système d'exploitation pour la cible, • objet résultat pour le détecteur, • objet du serveur d'applications pour la cible, • environnement de traitement pour la cible, • objet de départ (seed) ayant causé la génération de la cible par le moteur de reconnaissance.
getCsTarget	Renvoie les informations suivantes : • descripteur du système d'exploitation pour la cible, • objet résultat pour le détecteur, • objet du système informatique pour la cible, • environnement de traitement pour la cible, • objet de départ (seed) ayant causé la génération de la cible par le moteur de reconnaissance.
getComputerSystem	Renvoie un objet auquel le descripteur du système d'exploitation est connecté, les attributs étant renseignés.
getNewOsHandle	Tente de créer un nouveau descripteur de système d'exploitation pour la cible spécifiée. Peut servir à communiquer avec une machine différente de celle pour laquelle le détecteur personnalisé a été lancé à l'origine.
getOperatingSystem	Renvoie un objet représentant le système d'exploitation auquel est connecté le descripteur.
queryRegistry	Renvoie une clé de registre en langage XML.

Fonctions de chemin :

Les fonctions de chemin permettent de convertir des caractères de séparation de chemin Windows et Unix.

Vous pouvez utiliser les fonctions de chemin pour échanger des caractères séparateurs de chemin Microsoft Windows et Unix. Le tableau 38 décrit les fonctions que vous pouvez utiliser.

Tableau 38. Fonctions de chemin

Fonction	Description
unixSlashes	Permet de substituer des caractères séparateurs de chemin Windows à des séparateurs de chemin Unix.

Tableau 38. Fonctions de chemin (suite)

Fonction	Description
windowsSlashes	Permet de substituer des caractères séparateurs de chemin Unix à des séparateurs de chemin Windows.

Fonctions d'utilitaire :

Les fonctions d'utilitaire vous permettent d'initialiser l'interface de programme d'application des extensions de serveur personnalisé et d'exécuter diverses tâches utiles.

Vous pouvez utiliser les fonctions de programme utilitaire pour initialiser l'interface de programme d'application des extensions de serveur personnalisé. Vous pouvez créer un tableau dans Python à utiliser avec certaines fonctions Java et TADDM ainsi que le fractionnement des lignes de commandes dans leurs composants.

Le tableau 39 décrit les fonctions que vous pouvez utiliser.

Tableau 39. Fonctions d'utilitaire

Fonction	Description
findElementsForXPath	Recherche des objets et renvoie les résultats de la requête.
getArray	Permet d'obtenir un tableau dans Python à utiliser avec certaines fonctions Java et TADDM.
init	Permet d'initialiser l'interface de programme d'application d'extension de serveur personnalisé.
splitArgs	Permet de diviser une ligne de commande entre ses composants et de les renvoyer sous forme de séquence Python.

Fonctions d'informations de version :

Les fonctions d'informations de version permettent de déterminer les numéros de version de l'interface de programme d'application.

Vous pouvez utiliser les fonctions d'informations de version pour déterminer les numéros de version secondaire ou principale de l'interface de programme d'application des extensions de serveur personnalisé, ainsi que le numéro de version TADDM. Le tableau 40 décrit les fonctions que vous pouvez utiliser.

Tableau 40. Fonctions d'informations de version

Fonction	Description
getApiMinorVersion	Permet de renvoyer le numéro de la version secondaire de l'interface de programme d'application.
getApiVersion	Permet de renvoyer le numéro de la version principale de l'interface de programme d'application.

Tableau 40. Fonctions d'informations de version (suite)

Fonction	Description
getTADDMVersion	Permet de renvoyer le numéro de version TADDM.

Référence de fonction

Cette référence décrit chaque fonction disponible dans l'API des extensions de serveur personnalisé. Les fonctions sont répertoriées dans l'ordre alphabétique.

fonction addProcessToPool :

Ajoute un processus d'exécution à un pool de processus.

Description

La fonction addProcessToPool ajoute un objet RuntimeProcess Common Data Model (CDM) à un ProcessPool. (Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

addProcessToPool (*rp*, *pool*, **os*)

Paramètres

rp Objet RuntimeProcess du CDM

pool
Objet ProcessPool du CDM

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet du descripteur de système d'exploitation connecté à la nouvelle cible.

Exceptions

OsException.

fonction binToDot :

Permet de convertir une représentation binaire d'une adresse IP en notation décimale.

Description

La fonction binToDot permet de convertir une représentation binaire d'une adresse IP en notation décimale.

Syntaxe de fonction

binToDot (*binIp*)

Paramètres

binIp
Type long Python contenant la représentation binaire de l'adresse IP

Résultats

La fonction renvoie la représentation de chaîne d'une adresse de réseau IP en notation décimale.

Exceptions

Néant.

fonction bitsMaskToDottedDecimalMask :

Permet de convertir la notation de masque de contrôle des données du réseau en notation décimale avec point.

Description

La fonction bitsMaskToDottedDecimalMask permet de convertir la notation de masque de contrôle des données du réseau en représentation décimale avec point (par exemple, 24 > 255.255.255.0). Ne spécifiez pas la barre oblique de début (/) dans le masque de contrôle des données. Le nombre de bits correct est 8 et 16-32.

Syntaxe de fonction

bitsMaskToDottedDecimalMask (*bits*)

Paramètres

bits
Représentation de chaîne du nombre de contrôle des données

Résultats

La fonction renvoie le format décimal de l'adresse.

Exceptions

Néant.

fonction calcNetworkAddress :

Permet de calculer l'adresse réseau, à partir d'une adresse IP et d'un masque de réseau.

Description

La fonction calcNetworkAddress permet de calculer l'adresse réseau à partir de l'adresse IP et du masque de réseau spécifiés.

Syntaxe de fonction

calcNetworkAddress (*ip*)

Paramètres

ip Représentation de chaîne de l'adresse IP

(*mask*)
Représentation de chaîne du masque de sous-réseau

Résultats

La fonction renvoie la représentation de chaîne d'une adresse de réseau IP.

Exceptions

Néant.

fonction canonicalMac :

Permet de supprimer les séparateurs ou la notation de base d'une adresse MAC pour la renvoyer sous forme d'adresse MAC hexadécimale.

Description

La fonction canonicalMac permet de supprimer les séparateurs ou la notation de base d'une adresse MAC pour la renvoyer sous forme d'adresse MAC hexadécimale en tant que chaîne avec caractères alphanumériques en majuscules.

Syntaxe de fonction

canonicalMac (*mac*)

Paramètres

mac
Adresse MAC

Résultats

La fonction renvoie une représentation de chaîne de l'adresse MAC canonique.

Exceptions

Néant.

fonction classlessNotation :

Permet de calculer la notation sans classe d'un réseau IP.

Description

La fonction classlessNotation permet de calculer la notation sans classe d'un réseau IP.

Syntaxe de fonction

classlessNotation (*ip*, *mask*)

Paramètres

ip Représentation de chaîne du réseau IP

mask
Représentation de chaîne du masque de sous-réseau

Résultats

La fonction renvoie la notation sans classe sous forme de chaîne.

Exceptions

Néant.

fonction cloneModelObject :

Permet de créer une copie d'un ModelObject du Common Data Model (CDM).

Description

La fonction cloneModelObject permet de créer une copie d'un ModelObject du Common Data Model (CDM), renvoyé indéfiniment de façon récursive via n'importe quel attribut enfant de ModelObject.

Syntaxe de fonction

cloneModelObject (*mo*)

Paramètres

mo ModelObject du CDM à cloner

Résultats

La fonction renvoie le clone du ModelObject du CDM.

Exceptions

Néant.

fonction dotToBin :

Permet de convertir une adresse IPv4 d'une notation décimale en format binaire.

Description

La fonction dotToBin permet de convertir une adresse IPv4 d'une notation décimale en format binaire.

Syntaxe de fonction

dotToBin (*ip*)

Paramètres

ip Représentation de chaîne d'une adresse IP.

Résultats

La fonction renvoie l'adresse IP sous forme binaire en tant que type long Python.

Exceptions

NumberFormatException si l'adresse IP n'est pas valide.

fonction executeCommand :

Exécute une commande sur la cible.

Description

La fonction executeCommand exécute une commande sur la cible, selon le délai d'attente par défaut de deux minutes. Sachez que la fonction ajoute en préfixe le paramètre PATH pour le type de cible, comme spécifié dans le fichier collation.properties.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

executeCommand (*cmd*, **os*)

Paramètres

cmd
Chaîne de commande à exécuter

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie une chaîne contenant la sortie de la commande.

Exceptions

OsException.

fonction executeCommandWithTimeout :

Exécute une commande sur la cible avec un délai d'attente spécifiant la durée d'exécution autorisée pour la commande.

Description

La fonction executeCommandWithTimeout exécute une commande sur la cible, avec un délai d'attente spécifiant la durée d'exécution autorisée pour la commande. Sachez que la fonction ajoute en préfixe le paramètre PATH pour le type de cible, comme spécifié dans le fichier collation.properties.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des

cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

executeCommandWithTimeout (*cmd*, *timeout*, **os*)

Paramètres

cmd
Chaîne de commande à exécuter

timeout
Délai pendant lequel la commande est autorisée à s'exécuter (en millisecondes)

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie une chaîne contenant la sortie de la commande.

Exceptions

OsException.

fonction getApiMinorVersion :

Permet de renvoyer le numéro de la version secondaire de l'interface de programme d'application.

Description

La fonction getApiMinorVersion renvoie le numéro de la version secondaire de l'interface de programme d'application de l'extension de serveur personnalisé.

Syntaxe de fonction

getApMinoriVersion

Paramètres

Néant

Résultat

La fonction renvoie le numéro de la version secondaire de l'interface de programme d'application.

Exceptions

Néant.

fonction getApiVersion :

Permet de renvoyer le numéro de la version principale de l'interface de programme d'application.

Description

La fonction getApiVersion renvoie le numéro de la version principale de l'interface de programme d'application de l'extension de serveur personnalisé.

Syntaxe de fonction

getApiVersion

Paramètres

Néant

Résultat

La fonction renvoie le numéro de la version principale de l'interface de programme d'application.

Exceptions

Néant.

fonction getAppTarget :

Permet de renvoyer un bloc de données contenant des informations sur la cible de l'application.

Description

La fonction getAppTarget renvoie les informations suivantes :

- Descripteur du système d'exploitation pour la cible
- Objet résultat pour le détecteur
- Objet du serveur d'applications pour la cible
- Environnement de traitement pour la cible
- Objet de départ ayant causé la génération de la cible par le moteur de reconnaissance

Syntaxe de fonction

getAppTarget (*target*)

Paramètres

target
Mappe de la cible

Résultats

La fonction renvoie un bloc de données contenant le descripteur du système d'exploitation pour la cible, l'objet résultat, l'objet AppServer du CDM, l'environnement du processus (le cas échéant) et l'objet de départ (seed).

Exceptions

Néant.

fonction getArray :

Permet d'obtenir un tableau dans Python à utiliser avec certaines fonctions Java et TADDM.

Description

La fonction getArray renvoie un tableau dans Python à utiliser avec certaines fonctions Java et TADDM. La fonction utilise le module de tableau Jython pour créer le tableau.

Syntaxe de fonction

getArray (*seq*, *classname*)

Paramètres

seq
Liste ou séquence Python

classname
Nom de classe Java complet correspondant au type d'objets spécifié dans le paramètre **seq**

Résultats

La fonction renvoie un tableau Java approprié pour transmission à des méthodes Java requérant un tableau.

Exceptions

Néant.

fonction getComputerSystem :

Renvoie un objet auquel le descripteur du système d'exploitation est connecté, les attributs étant renseignés.

Description

La fonction getComputerSystem permet de renvoyer l'objet ComputerSystem du Common Data Model (CDM) auquel le descripteur du système d'exploitation est connecté, les attributs étant renseignés.

Syntaxe de fonction

getComputerSystem (**os*)

Paramètres

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet ComputerSystem du CDM.

Exceptions

OsException.

fonction getCsTarget :

Permet de renvoyer un bloc de données contenant des informations sur la cible.

Description

La fonction getCsTarget renvoie un bloc de données contenant les informations suivantes :

- Descripteur du système d'exploitation pour la cible
- Objet résultat pour le détecteur
- Objet du système informatique pour la cible
- Environnement de traitement pour la cible
- Objet de départ ayant causé la génération de la cible par le moteur de reconnaissance

Syntaxe de fonction

getCsTarget (*target*)

Paramètres

target
Mappe de cible transmise à l'extension du serveur personnalisé.

Résultats

La fonction renvoie un bloc de données contenant le descripteur du système d'exploitation pour la cible, l'objet résultat, l'objet ComputerSystem du CDM et l'objet de départ (seed).

Exceptions

Néant.

fonction getFile :

Permet de capturer un fichier à partir d'une cible distante, et de renvoyer les métadonnées et le contenu du fichier.

Description

La fonction getFile capture un fichier à partir de la cible distante et renvoie un objet FileSystemContent du Common Data Model (CDM) contenant les métadonnées et le contenu du fichier.

Syntaxe de fonction

getFile (*path*, **os*)

Paramètres

path
Chemin du fichier à capturer

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet FileSystemContent du CDM contenant les métadonnées et le contenu du fichier.

Exceptions

OsException.

fonction getFileWithLengthLimit :

Capture un fichier, jusqu'à la longueur maximale spécifiée, à partir d'une cible distante, et renvoie les métadonnées et le contenu du fichier.

Description

La fonction getFileWithLengthLimit capture un fichier à partir de la cible distante et renvoie un objet FileSystemContent du Common Data Model (CDM) contenant les métadonnées et le contenu du fichier.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getFileWithLengthLimit (*path*, *length*, **os*)

Paramètres

path
Chemin du fichier à capturer

length
Longueur maximale à capturer (en octets)

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie un objet FileSystemContent du CDM contenant les métadonnées et le contenu du fichier.

Exceptions

OsException.

fonction getLocalDNSLookup :

Effectuez une recherche de nom d'hôte sur le serveur TADDM.

Description

La fonction getLocalDNSLookup effectue une recherche de nom sur le serveur TADDM et renvoie un objet DNSLookup du Common Data Model (CDM) contenant le résultat. La fonction accepte également un descripteur de système d'exploitation facultatif, mais, la recherche reste toujours locale.

Syntaxe de fonction

getLocalDNSLookup (*name*, **os*)

Paramètres

name
Nom à résoudre

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet DNSLookup du CDM.

Exceptions

OsException.

fonction getNewOsHandle :

Permet de créer un nouveau descripteur de système d'exploitation pour la cible spécifiée.

Description

La fonction getNewOsHandle tente de créer un nouveau descripteur de système d'exploitation pour la cible spécifiée. Vous pouvez l'utiliser pour communiquer avec une machine différente de celle pour laquelle le détecteur personnalisé a été lancé à l'origine. Une exception est émise s'il est impossible d'établir une session SSH ou WMI à l'aide des listes d'accès actuellement configurées dans le serveur TADDM.

Syntaxe de fonction

getNewOsHandle (*ip*)

Paramètres

ip Adresse IP de la machine à laquelle vous souhaitez vous connecter

Résultats

La fonction renvoie l'objet du descripteur de système d'exploitation connecté à la nouvelle cible.

Exceptions

OsException.

fonction getOperatingSystem :

Renvoie un objet représentant le système d'exploitation auquel est connecté le descripteur.

Description

La fonction getOperatingSystem permet de renvoyer l'objet OperatingSystem du Common Data Model (CDM) représentant le système d'exploitation auquel est connecté le descripteur.

Syntaxe de fonction

getOperatingSystem (**os*)

Paramètres

**os*
Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet OperatingSystem du CDM.

Exceptions

OsException.

fonction getPidConnectionMap :

Renvoie un dictionnaire Python contenant les ID de processus et les connexions TCP des ID de processus.

Description

La fonction getPidConnectionMap renvoie un dictionnaire Python des listes Python contenant les ID de processus comme clés, et des listes des connexions TCP des ID de processus.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getPidConnectionMap ( )

Paramètres

Néant.

Résultats

La fonction renvoie un dictionnaire Python de listes Python contenant les informations suivantes :

- ID de processus

- Connexions TCP des ID de processus

Exceptions

Néant.

fonction getPidPortList :

Renvoie un dictionnaire Python contenant les ID de processus et les ports qu'utilise le processus pour l'écoute ou la connexion.

Description

La fonction getPidPortList renvoie un dictionnaire Python des listes Python contenant les ID de processus comme clés, et des listes de ports qu'utilise le processus pour l'écoute ou la connexion.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getPidPortList (**os*)

Paramètres

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie un dictionnaire Python contenant les informations suivantes :
- ID de processus
- Listes Python des objets BindAddress du CDM

Exceptions

OsException.

fonction getPidToRuntimeProcessMap :

Renvoie un dictionnaire Python contenant les ID de processus et les informations sur le processus d'exécution.

Description

La fonction getPidToRuntimeProcessMap renvoie un dictionnaire Python avec les clés contenant les ID de processus et les valeurs représentant les objets RuntimeProcess du Common Data Model (CDM). (Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getPidToRuntimeProcessMap (**os*)

Paramètres

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie un dictionnaire Python contenant les informations suivantes :

- ID de processus
- Objets RuntimeProcess du CDM

Exceptions

Néant.

fonction getProcessByPid :

Permet de renvoyer l'objet RuntimeProcess du Common Data Model (CDM) associé à un ID de processus donné.

Description

La fonction getProcessByPid renvoie l'objet RuntimeProcess Common Data Model (CDM) associé à l'ID de processus spécifié. (Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getProcessByPid (*pid*, **os*)

Paramètres

pid
ID de processus.

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie l'objet RuntimeProcess du CDM ou "Néant" si l'ID de processus n'existe pas.

Exceptions

Néant.

fonction getRemoteDNSLookup :

Permet de rechercher un nom extrait d'une configuration distante qui risque de ne pas se résoudre sur le serveur TADDM.

Description

La fonction getLocalDNSLookup effectue une recherche de nom sur le système spécifié dans le premier paramètre. Vous pouvez utiliser cette fonction pour résoudre des noms extraits de configurations distantes, susceptibles de ne pas se résoudre sur le serveur TADDM.

Syntaxe de fonction

getRemoteDNSLookup (*ip*, *name*)

Paramètres

ip Adresse IP de la machine où doit se produire la recherche

name
Nom à résoudre

Résultats

La fonction renvoie l'objet DNSLookup du CDM.

Exceptions

OsException.

fonction getServerProcesses :

Renvoie un dictionnaire Python de listes Python contenant les ID de processus et les adresses de liaison des ports d'écoute.

Description

La fonction getServerProcesses renvoie un dictionnaire Python de listes Python d'objets BindAddress Common Data Model (CDM). (Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getServerProcesses (**os*)

Paramètres

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

Renvoie un dictionnaire Python contenant les informations suivantes :

- ID de processus
- Objets BindAddress de CDM pour les ports d'écoute des ID de processus

Exceptions

OsException.

Fonction getSimpleCapabilitiesFactory :

Renvoie un objet SimpleCapabilitiesFactory pour une adresse IP donnée.

Description

La fonction getSimpleCapabilitiesFactory permet de renvoyer une valeur SimpleCapabilitiesFactory pour une adresse IP donnée. SimpleCapabilitiesFactory peut servir à extraire les fonctionnalités suivantes :

ExecuteCapability
Cette fonctionnalité permet à une commande de s'exécuter sur une cible donnée, quel que soit le protocole de communication utilisé.

MibQueryCapability
Cette fonctionnalité permet à une requête MIB de s'exécuter sur une cible donnée.

OsInfoCapability
Cette fonctionnalité permet d'extraire des informations sur le système d'exploitation, sur une cible donnée.

Pour plus d'informations sur ces fonctionnalités, voir la documentation Javadoc décrite dans «Information JavadocTADDM», à la page 189.

Syntaxe de fonction

getSimpleCapabilitiesFactory (*ip*)

Paramètres

ip Objet IpV4Address représentant l'adresse IP de l'hôte cible.

Résultat

La fonction renvoie un objet SimpleCapabilitiesFactory.

Exceptions

IllegalArgumentException si le paramètre IP a la valeur null ou ne correspond pas à une adresse IPv4 valide.

fonction getTADDMVersion :

Permet de renvoyer le numéro de version TADDM.

Description

La fonction getTADDMVersion renvoie la version de TADDM utilisé.

Syntaxe de fonction

getTADDMVersion

Paramètres

Néant

Résultats

La fonction renvoie la version de TADDM utilisée.

Exceptions

Néant.

fonction init :

Permet d'initialiser l'interface de programme d'application d'extension de serveur personnalisé.

Description

La fonction init permet d'initialiser les routines auxiliaires de l'interface de programme d'application de l'extension de serveur personnalisé.

Syntaxe de fonction

init (*target*)

Paramètres

target
Mappe des cibles

Résultats

La fonction renvoie un bloc de données contenant les informations suivantes :

- Descripteur du système d'exploitation pour la cible
- Objet résultat
- Objet ComputerSystem ou AppServer du Common Data Model (CDM)
- Objet de départ (seed)
- Le journal d'événements permettant d'écrire dans l'environnement du journal du détecteur (si la cible est un objet AppServer)

Exceptions

Néant.

fonction ipInSubnet :

Permet de déterminer si une adresse IP est un membre d'un sous-réseau donné et non l'adresse de diffusion.

Description

La fonction ipInSubnet permet de déterminer si une adresse IP est un membre d'un sous-réseau donné et non l'adresse de diffusion.

Syntaxe de fonction

ipInSubnet (*ip*, *net*, *mask*)

Paramètres

ip Représentation de chaîne d'une adresse IP.

net
Représentation de chaîne d'un réseau

mask
Masque de sous-réseau du réseau

Résultats

La fonction renvoie les informations suivantes :

- Valeur différente de zéro si l'adresse IP est membre du sous-réseau
- 0 si l'adresse IP n'est pas membre du sous-réseau

Exceptions

Néant.

fonction listDirectory :

Permet de renvoyer une liste Python présentant le contenu d'un répertoire sur une cible distante.

Description

La fonction listDirectory renvoie une liste Python présentant le contenu d'un répertoire sur une cible distante.

Syntaxe de fonction

listDirectory (*path, *os*)

Paramètres

path
Chemin d'accès au répertoire

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie la séquence Python du contenu du répertoire.

Exceptions

OsException.

fonction networkToList :

Permet de renvoyer toutes les adresses IP qui sont membres de la représentation CDM du paramètre IpNetwork.

Description

La fonction networkToList permet de renvoyer toutes les adresses IP qui sont membres de la représentation CDM du paramètre IpNetwork.

Syntaxe de fonction

networkToList (*net*)

Paramètres

net
Objet IpNetwork du Common Data Model (CDM)

Résultats

La fonction renvoie une liste Python de représentations de chaîne des adresses IP.

Exceptions

Néant.

fonction newModelObject :

Permet de créer un objet Common Data Model (CDM).

Description

La fonction newModelObject crée un nouvel objet de modèle du type de classe Common Data Model (CDM).

Syntaxe de fonction

newModelObject (*classname*)

Paramètres

classname
Nom de classe CDM complet du ModelObject CDM à créer

Résultats

La fonction renvoie le nouveau ModelObject du CDM.

Exceptions

Néant.

fonction queryRegistry :

Renvoie une clé de registre en langage XML.

Description

La fonction queryRegistry renvoie la clé de registre demandée en langage XML. Cette fonction est opérationnelle uniquement si le descripteur du système d'exploitation est connecté à une cible Windows ; sinon, une exception est renvoyée.

(Facultatif) Vous pouvez transmettre un descripteur de système d'exploitation à la fonction. Sinon, le descripteur de système d'exploitation transmis à la mappe des cibles à la fonction init est utilisé par défaut.

Syntaxe de fonction

getOperatingSystem (*key*, **os*)

Paramètres

key
Clé de registre à extraire

**os*
(Facultatif) Objet du descripteur de système d'exploitation

Résultats

La fonction renvoie la représentation XML de la clé de registre

Exceptions

OsException et MethodNotImplementedException.

fonction setExtendedAttributes :

Définit les valeurs des attributs étendus.

Description

La fonction setExtendedAttributes accepte un ModelObject Common Data Model (CDM) et un dictionnaire Python de paires valeur-nom, et définit ces paires comme attributs étendus de ModelObject.

Syntaxe de fonction

setExtendedAttributes (*mo*, *exattrs*)

Paramètres

mo ModelObject CDM

exattrs
Dictionnaire Python de paires valeur-nom où le nom est celui de l'attribut étendu et la valeur, une chaîne

Exceptions

IoException.

fonction splitArgs :

Permet de diviser une ligne de commande entre ses composants et de les renvoyer sous forme de séquence Python.

Description

La fonction splitArs permet de diviser une ligne de commande entre ses composants et de les renvoyer sous forme de séquence Python.

Syntaxe de fonction

splitArgs (*cmdline*)

Paramètres

cmdline
Une ligne de commande (le paramètre doit être cité s'il contient des espaces imbriqués)

Résultats

La fonction renvoie une séquence Python contenant les jetons de ligne de commande.

Exceptions

Néant.

fonction unixSlashes :

Convertit les caractères de séparation de chemin Windows en caractères de séparation de chemin UNIX.

Description

La fonction unixSlashes convertit les caractères de séparation de chemin Windows en caractères de séparation de chemin UNIX.

Syntaxe de fonction

unixSlashes (*path*)

Paramètres

path
Chemin de système de fichiers susceptible de contenir des séparateurs de chemin Windows

Résultats

La fonction renvoie un chemin de système de fichiers contenant uniquement des séparateurs de chemin UNIX.

Exceptions

Néant.

fonction validateFqdn :

Permet de vérifier un nom de domaine complet pour garantir sa conformité aux règles décrites dans la RFC 1035.

Description

La fonction validateFqdn permet de vérifier un nom de domaine complet pour garantir sa conformité aux règles décrites dans la RFC 1035. Sachez que la console de gestion Discovery n'affiche pas les noms de domaine complets qui ne sont pas conformes.

Syntaxe de fonction

validateFqdn (*fqdn*)

Paramètres

fqdn
Nom de domaine complet

Résultats

La fonction renvoie les valeurs suivantes :

- Valeur différente de zéro si le nom de domaine complet est valide
- 0 si le nom de domaine complet n'est pas valide

Exceptions

Néant.

fonction validateIp :

Permet de valider une adresse IP en notation décimale.

Description

La fonction validateIp confirme qu'une adresse IP en notation décimale est valide.

Syntaxe de fonction

validateIp (*ip*)

Paramètres

ip Représentation de chaîne d'une adresse IP en notation décimale

Résultats

La fonction renvoie les valeurs suivantes :

- Valeur différente de zéro si l'adresse IP est valide
- 0 si l'adresse IP n'est pas valide

Exceptions

Néant.

fonction windowsSlashes :

Convertit les caractères de séparation de chemin UNIX en caractères de séparation de chemin Windows.

Description

La fonction windowsSlashes convertit les caractères de séparation de chemin UNIX en caractères de séparation de chemin Windows.

Syntaxe de fonction

windowsSlashes (*path*)

Paramètres

path
Chemin de système de fichiers susceptible de contenir des séparateurs de chemin UNIX

Résultats

La fonction renvoie un chemin de système de fichiers contenant uniquement des séparateurs de chemin Windows.

Exceptions

Néant.

Exemple d'application d'extension de serveur personnalisé

Une application classique d'extension de serveur personnalisé inclut plusieurs segments de code courants.

L'exemple suivant d'application d'extension de serveur personnalisé montre les éléments standard et les segments de code que vous pouvez inclure dans vos applications :

```
import sys
import java

from java.lang import System
coll_home = System.getProperty("com.collation.home")

System.setProperty("jython.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")
System.setProperty("python.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")

jython_home = System.getProperty("jython.home")
sys.path.append(jython_home + "/Lib")
sys.path.append(coll_home + "/lib/sensor-tools")
sys.prefix = jython_home + "/Lib"

import traceback
import string
import re
import jarray
import sensorhelper

############################
# LogError      Error logger
```

```
##########################
def LogError(msg):
        log.error(msg)
        (ErrorType, ErrorValue, ErrorTB) = sys.exc_info()
        traceback.print_exc(ErrorTB)

##########################
# main
##########################

try:
 (os_handle, result, appserver,seed,log,env) = sensorhelper.init(targets)

 response = sensorhelper.executeCommand("ssh -V 2>&1")

 if response != None:
  match = re.search("OpenSSH_([^,]+)",response)

  if match != None:
   appserver.setProductVersion(match.group(1))
   appserver.setProductName("OpenSSH")
   appserver.setVendorName("openssh.org")
  else:
   log.info("This ssh server does not appear to be OpenSSH")
 else:
    log.info("'ssh -V' returned no output")
except:
 LogError("unexpected exception getting ssh information")
```

Explication des segments dans l'exemple d'application

Cette section décrit des segments de l'exemple d'application d'extension de serveur personnalisé.

Initialisation de l'environnement

Cette section du code configure l'environnement afin que l'interpréteur Jython trouve les modules Python standard ainsi que le module Python sensortools de TADDM.

```
from java.lang import System
coll_home = System.getProperty("com.collation.home")

System.setProperty("jython.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")
System.setProperty("python.home",coll_home +
"/osgi/plugins/com.ibm.cdb.core.jython_1.0.0/lib")

jython_home = System.getProperty("jython.home")
sys.path.append(jython_home + "/Lib")
sys.path.append(coll_home + "/lib/sensor-tools")
sys.prefix = jython_home + "/Lib"
```

Importation de sensorhelper

Cette section du code importe le module Python sensortools de TADDM. Ce code permet à l'application d'appeler les fonctions d'extension de serveur personnalisé ; par exemple, **sensorhelper.executeCommand**("echo hello world").

```
import sensorhelper
```

Erreurs de consignation

Cette section de code consigne les traces exception stack à l'aide du module Python traceback. Pour une consignation suivie, vous pouvez utiliser l'objet journal renvoyé par l'appel sensorhelper.init().

```
def LogError(msg):
        log.error(msg)
        (ErrorType, ErrorValue, ErrorTB) = sys.exc_info()
        traceback.print_exc(ErrorTB)
```

Initialisation de sensorhelper

Cette section du code initialise le module Python à l'aide d'informations sur la cible de la reconnaissance transmises à l'application de l'extension de serveur personnalisé par le moteur de reconnaissance TADDM.

```
(os_handle, result, appserver,seed,log,env) = sensorhelper.init(targets)
```

Exécution de la commande

Cette section du code appelle la fonction **executeCommand()** de sensortools pour exécuter "`ssh –V`" sur la cible de reconnaissance. La sortie résultante de la commande est enregistrée dans la variable de la réponse.

En général, une extension de serveur personnalisé doit exécuter une ou plusieurs commandes et/ou capturer un ou plusieurs fichiers pour découvrir la cible visée.

```
response = sensorhelper.executeCommand("ssh -V 2>&1")
```

Recherche de réponse

Cette section du code vérifie d'abord la variable de réponse pour garantir sa validité (par exemple, que la valeur n'est pas None). Le code utilise ensuite le module d'expression régulière Python pour analyser la sortie de la commande `ssh –V` stockée dans la variable de réponse.

```
if response != None:
   match = re.search("OpenSSH_([^,]+)",response)
```

Définition des attributs

Cette section de code vérifie d'abord que la variable de réponse a été analysée avec succès par le module d'expression régulière Python (la valeur de la variable match n'est pas égale à None). Si la variable de réponse a été analysée avec succès, la version ssh est enregistrée dans l'attribut productVersion de l'objet AppServer du Common Data Model (CDM). En outre, les attributs productName et vendorName sont définis.

```
if match != None:
   appserver.setProductVersion(match.group(1))
   appserver.setProductName("OpenSSH")
   appserver.setVendorName("openssh.org")
```

Définition de l'objet

Cette section de code enregistre l'objet CDM AppServer dans l'objet Result. Une fois que le détecteur prend fin, tous les ModelObjects du CDM contenus dans l'objet Result sont envoyés dans le moteur de stockage TADDM pour être conservés dans la base de données.

```
result.setAppServer(appserver)
```

Recommandations pour le développement d'applications d'extensions de serveur personnalisé

Vous pouvez optimiser votre application d'extension de serveur personnalisé en respectant quelques règles simples.

Suivez ces instructions lors du développement d'applications d'extension de serveur personnalisé :

- Consignez les opérations exécutées par l'application.

  Vous pouvez utiliser l'objet journal renvoyé par la fonction `sensorhelper.init()` afin d'exécuter les opérations de consignation.
- Utilisez le module Python traceback pour consigner les traces d'exception stack.

Si vous utilisez le fichier `sensorstub.py` dans le répertoire `dist/lib/sensor-tools` comme base pour votre application d'extension de serveur personnalisé, la fonction `LogError` définie dans le fichier exécute cette tâche.

- L'augmentation du nombre d'objets de modèle dans l'objet de résultat entraîne celle du temps nécessaire au stockage des objets.

Vues et schéma de base de données TADDM

La base de données TADDM contient tous les éléments de configuration gérés par un domaine TADDM.

Il est possible de renseigner la base de données avec des éléments de configuration de différentes façons :

- Reconnaissances TADDM
- Chargement en bloc de fichiers de livres de l'adaptateur de bibliothèque de reconnaissance
- Ajout manuel d'EC à l'aide de l'interface graphique
- Ajout par programmation d'éléments de configuration via l'interface de programmation TADDM

Les éléments de configuration de la base de données TADDM sont organisés en fonction de leur classification dans le Common Data Model (CDM) IBM Tivoli. Les types d'objet, attributs, relations et règles de nommage du CDM sont documentés dans le fichier CDMWebsite.zip, situés dans le répertoire `$COLLATION_HOME/sdk/doc/model`. Pour parcourir la documentation, décompressez le fichier .zip dans un nouveau répertoire, puis ouvrez le fichier `misc/CDM.htm` dans un navigateur Web.

Pour plus d'informations sur le Common Data Model Tivoli, voir «Présentation du Common Data Model», à la page 1.

Lorsque vous développez des rapports personnalisés pour TADDM, vous pouvez utiliser des requêtes SQL pour récupérer les données stockées dans la base de données TADDM. Cependant, au lieu d'lancer une requête directement sur des données dans les tables de la base de données TADDM, les rapports doivent utiliser les vues de la base de données TADDM. TADDM fournit de nombreuses vues de base de données prédéfinies pour simplifier la tâche d'écriture de requêtes SQL demandant l'extraction de données de la base de données et vous pouvez également concevoir vos propres vues personnalisées.

Il existe quatre catégories de vues de base de données TADDM :

- Vues de bloc fonctionnel
- Vues du panneau Détails
- Vues personnalisées
- Vues des attributs étendus

Remarque : Fix Pack 3 Dans les versions TADDM 7.3.0.3 et ultérieures, le nombre maximal de caractères pour les noms de colonnes dans les vues de base de données est 30.

Concepts associés:

«Dictionnaire de données TADDM», à la page 187
Le dictionnaire de données TADDM est un ensemble de pages HTML générées automatiquement permettant de mapper les informations du Common Data Model à celles de la base de données TADDM.

Vues de bloc fonctionnel

Vous pouvez utiliser les vues de bloc de construction pour écrire des requêtes basées sur le point de vue du Common Data Model (CDM) Tivoli. Ces vues sont utiles si vous êtes familier avec le CDM ; elles n'exigent aucune connaissance de l'emplacement de stockage des éléments de configuration dans les tables de la base de données TADDM.

Pour consulter la documentation détaillée relative aux vues de bloc de construction, accédez au répertoire `$COLLATION_HOME/etc/views` et ouvrez l'un des fichiers suivants :

- Pour les bases de données DB2 : `create_building_block_views_db2.sql`
- Pour les bases de données Oracle : `create_building_block_views_oracle.sql`

Les commentaires de ces fichiers décrivent les types d'objet Common Data Model et les vues de base de données correspondantes, en indiquant les mappages entre le Common Data Model et le schéma de base de données :

- entre un type d'objet CDM et un nom de vue de la base de données du bloc fonctionnel,
- entre un attribut et un nom de colonne de vue de la base de données,
- entre une relation et la syntaxe JOIN.

Le nom de chaque vue de bloc fonctionnel se présente sous la forme suivante :
`BB_%_V`

Dans chaque nom de vue, le symbole `%` représente le nom du type d'objet. Chaque type d'objet est mappé à une vue du bloc fonctionnel, résultant en plus de 1000 vues de base de données de bloc fonctionnel représentant des types d'objet.

Par exemple, le CDM définit le type d'objet `sys.windows.WindowsComputerSystem`, qui est enregistré dans le tableau de base COMPSYS de la base de données TADDM. Ce tableau contient également de nombreux autres types d'objet étendant le type d'objet `sys.ComputerSystem` (par exemple, `sys.LinuxUnitaryComputerSystem`).

Pour rechercher des éléments de configuration `sys.windows.WindowsComputerSystem` à l'aide d'une vue de bloc fonctionnel, interrogez la vue de la base de données BB_WINDOWSCOMPUTERSYSTEM20_V.

Outre les vues représentant les types d'objet, plus de 800 vues spéciales prennent en charge les relations "plusieurs-à-plusieurs" entre types d'objet. Chacune de ces vues de bloc fonctionnel pour table de mappage porte un nom du format suivant :
`BB_%J`

Pour plus d'informations sur ces vues spéciales, voir «Définitions de JOIN», à la page 172.

Définitions de vue

Pour chaque vue de bloc fonctionnel, les commentaires incluent une section décrivant la vue et les objets CDM correspondants. Par exemple, la section de définition de la vue de bloc fonctionnel BB_WINDOWSCOMPUTERSYSTEM20_V apparaît comme suit :

```
-- ######## model.topology.sys.windows.WindowsComputerSystem ########
--
-- View..... BB_WINDOWSCOMPUTERSYSTEM20_V
-- Class..................... model.topology.sys.windows.WindowsComputerSystem
-- Super classes................ model.topology.sys.ComputerSystem
-- model.topology.core.ManagedElement
-- model.ModelObject
-- model.topology.process.itil.ConfigurationItem
```

Cet exemple montre que la vue BB_WINDOWSCOMPUTERSYSTEM20_V correspond au type d'objet `model.topology.sys.WindowsComputerSystem` et répertorie plusieurs superclasses dont ce type hérite des attributs et des définitions de relation.

Définitions de colonne pour les attributs

Pour chaque colonne correspondant à un attribut CDM, les commentaires incluent une section décrivant la colonne et l'attribut correspondant. Par exemple, la section de définition de colonne de la vue CPUSPEED_C column of the BB_WINDOWSCOMPUTERSYSTEM20_V apparaît comme suit :

```
-- Column.... CPUSPEED_C
-- Attribute..................... CPUSpeed
-- Java Type..................... long, primitive
-- Declared By................... model.topology.sys.ComputerSystem
```

Cet exemple montre que la colonne CPUSPEED_C correspond à l'attribut `CPUSpeed` défini par le type d'objet `model.topology.sys.ComputerSystem`. Cet attribut est hérité par le type `model.topology.sys.WindowsComputerSystem`. Il répertorie également le type Java utilisé pour représenter la valeur de l'attribut.

Remarque : Les commentaires n'indiquent pas la liste des types de base de données utilisés pour stocker l'attribut. Vous pouvez déterminer le type de base de données en utilisant une commande SQL **`describe`** : La commande describe renvoie plusieurs colonnes qui incluent les noms des vues de base de données, que vous pouvez utiliser pour interroger les données affichées dans l'onglet **Panneau Détails**.

```
db2 describe table BB_WindowsComputerSystem20_V
```

Définitions de colonne pour les relations [0..1]

Pour chaque colonne représentant une relation CDM de type "zéro-à-un" pour un autre élément de configuration, les commentaires incluent une section décrivant la colonne utilisée pour exécuter l'opération SQL JOIN pour l'élément de configuration de l'autre côté de la relation. Par exemple, la section de définition de la colonne PK_OSRUNNING_C de la vue BB_WINDOWSCOMPUTERSYSTEM20_V apparaît comme suit :

```
-- Column.... PK__OSRUNNING_C
-- Attribute..................... OSRunning
-- Java Type..................... model.topology.sys.OperatingSystem, notContained
-- Declared By................... model.topology.sys.ComputerSystem
```

Cet exemple montre que la colonne PK__OSRUNNING_C sert à exécuter l'opération JOIN sur le tableau OperatingSystem, qui représente la relation `OSRunning` définie par le type d'objet `model.topology.sys.ComputerSystem`. Il indique également que le typeJava de la valeur d'attribut est, dans ce cas, un autre type CDM (`model.topology.sys.OperatingSystem`).

Remarque : Toutes les colonnes qui représentent des relations avec d'autres éléments de configuration (et ne sont donc pas des types de primitive) portent des noms commençant par `PK__`. D'un point de vue de base de données relationnelle, la valeur d'une colonne de ce type est l'identificateur global unique de l'élément de configuration se trouvant de l'autre côté de la relation.

Fix Pack 3

Colonne d'horodatage dans les versions TADDM 7.3.0.3 et ultérieures

Dans les versions TADDM 7.3.0.3 et ultérieures, les vues de bloc fonctionnel affichent une colonne d'horodatage supplémentaire. Par exemple, la section de définition de la vue de la colonne LASTSTOREDTIME_C de la vue BB_WINDOWSCOMPUTERSYSTEM20_V comporte une chaîne `(timestamp)` supplémentaire :

```
-- Column.... LASTSTOREDTIME_C
-- Attribute..................... lastStoredTime
-- Java Type..................... long, primitive (timestamp)
-- Declared By................... model.ModelObject
```

Chaque attribut Horodatage contient les deux colonnes suivantes :

- LASTSTOREDTIME_C, définie sur le type `long`, par exemple `1445417251307`.
- LASTSTOREDTIME_T, contenant l'horodatage lisible par l'utilisateur, par exemple `Oct 21, 2015 10:47:31 AM`.

Comme pour la convention de dénomination initiale de la colonne d'horodatage, la colonne ajoutée a toujours le même suffixe, à savoir dans ce cas `_T`.

Définitions de JOIN

Pour les deux relations CDM de type "zéro-à-un" et "plusieurs-à-plusieurs", les commentaires fournissent un exemple de requête SQL indiquant comment accomplir l'opération SQL JOIN. Par exemple, la section de définition JOIN de la relation `OSRunning` de la vue BB_WINDOWSCOMPUTERSYSTEM20_V apparaît comme suit :

```
-- Join......
-- Attribute..................... OSRunning
-- Java Type..................... model.topology.sys.OperatingSystem, notContained
-- Declared By................... model.topology.sys.ComputerSystem
-- Test Join..................... SELECT COUNT(1) FROM
--                                BB_WINDOWSCOMPUTERSYSTEM20_V T1,
--                                BB_OPERATINGSYSTEM62_V T2
--                                WHERE T1.PK__OSRUNNING_C = T2.PK_C
```

Certaines relations entre éléments de configuration sont du type "plusieurs-à-plusieurs" ; par exemple, un élément de configuration `sys.windows.WindowsComputerSystem` pourrait avoir une relation avec plusieurs éléments de configuration `sys.FileSystem`, représentée par une relation `contains` utilisant l'attribut `fileSystems` de `sys.windows.WindowsComputerSystem` :

```
-- Join......
-- Attribute..................... fileSystems
-- Java Type..................... Array of model.topology.sys.FileSystem, array
-- Declared By................... model.topology.sys.ComputerSystem
-- Test Join..................... SELECT COUNT(1) FROM
--                                BB_WINDOWSCOMPUTERSYSTEM20_V T1,
--                                BB_COMPUTERSYSTTEMS_88841D4BJ T2,
--                                BB_FILESYSTEM71_V T3
--                                WHERE T1.PK_C = T2.PK__JDOID_C
--                                AND T3.PK_C = T2.PK__FILESYSTEMS_C
```

Les relations "plusieurs-à-plusieurs" sont enregistrées dans la table de mappage intermédiaire (dans cet exemple, la table de mappage BB_COMPUTERSYSTTEMS_88841D4BJ).

Vues obsolètes

Certaines vues de bloc de construction deviennent dépréciées suite à des modifications du Common Data Model (CDM) de Tivoli et seront supprimées à l'avenir. Reportez-vous au tableau ci-dessous qui répertorie les vues dépréciées et leurs équivalents afin d'effectuer les modifications nécessaires.

Pour transformer les rapports qui sont déjà définis en travaux après la mise à niveau, les affichages de compatibilité sont définis et marqués comme dépréciés.

Si vous utilisez une des vues répertoriées dans le tableau suivant, corrigez votre définition de rapport afin d'utiliser une nouvelle définition de vue équivalente.

Le tableau suivant répertorie les vues obsolètes qui prennent en charge les relations "plusieurs-à-plusieurs" entre les types d'objet et leurs nouveaux équivalents.

Tableau 41. Les vues dépréciées et leurs nouveaux équivalents.

Vue obsolète	Nouvel équivalent
BB_APPCONFIGJDONCES_FCB57E09J	BB_SPHYSICALFILNCES_ECF04BA6J
BB_APPCONFIGJDO_ROLES_J	BB_SPHYSICALFILEJDO_ROLES_J
BB_APPSERVERCLUNCES_E03E73D6J	BB_SGROUPJDO_SENCES_CF68C060J
BB_APPSERVERCLUOLES_F1CF6FA2J	BB_SGROUPJDO_ROLES_J
BB_APPSERVERJDONCES_EF1C0CA8J	BB_SSOFTWARESERNCES_19E863EFJ
BB_APPSERVERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_COLLECTIONJDNCES_2AB18ECEJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_COLLECTIONJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_COMPOSITEJDONCES_C1981AC5J	BB_SGROUPJDO_SENCES_CF68C060J
BB_COMPOSITEJDO_MEMBERS_J	BB_COLLECTIONJDO_MEMBERS_J
BB_COMPOSITEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_COMPUTERSYSTEMJDO_ROLES_J	BB_SCOMPUTERSYSTEMJDO_ROLES_J
BB_COMPUTERSYSTNCES_611B3FC2J	BB_SCOMPUTERSYSNCES_119A868FJ
BB_COMPUTERSYSTNCES_6A2E6F7CJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_COMPUTERSYSTOLES_6B483FBCJ	BB_SGROUPJDO_ROLES_J
BB_CONFIGURATIONCES_F7B28A51J	BB_SFUNCTIONJDONCES_F8394E61J
BB_CONFIGURATIOOLES_33A89807J	BB_SFUNCTIONJDO_ROLES_J
BB_DB2DATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J

Tableau 41. Les vues dépréciées et leurs nouveaux équivalents. (suite)

Vue obsolète	Nouvel équivalent
BB_DB2DATABASEJNCES_83C5253DJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_DB2SYSTEMJDONCES_6CAED009J	BB_SSOFTWAREINSNCES_549D08D8J
BB_DB2SYSTEMJDO_ROLES_J	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_DOMINOCONNECNCES_FAE09606J	BB_SPHYSICALFILNCES_ECF04BA6J
BB_DOMINOCONNECOLES_F12CCB72J	BB_SPHYSICALFILEJDO_ROLES_J
BB_DOMINODATABANCES_BE00AD89J	BB_SDEPLOYABLECNCES_572F3E83J
BB_DOMINODATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_DOMINODOMAINJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_DOMINODOMAINNCES_AC5777E0J	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEADMINCES_6B41ACDEJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEADMIOLES_5F958B9AJ	BB_SGROUPJDO_ROLES_J
BB_EXCHANGEFOLDNCES_EC2EEA5DJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEFOLDOLES_B3A7C77BJ	BB_SGROUPJDO_ROLES_J
BB_EXCHANGEMAILNCES_2B5725CJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_EXCHANGEMAILOLES_250648DCJ	BB_SGROUPJDO_ROLES_J
BB_FABRICJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_FABRICJDO_SENCES_783F8B27J	BB_SGROUPJDO_SENCES_CF68C060J
BB_FUNCTIONJDO_NCES_C03DA9D4J	BB_SFUNCTIONJDONCES_F8394E61J
BB_FUNCTIONJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_HACMPAPPRESONCES_B47F5A8AJ	BB_SFUNCTIONJDONCES_F8394E61J
BB_HACMPAPPRESOOLES_229BB16EJ	BB_SFUNCTIONJDO_ROLES_J
BB_HACMPLOCALREENTS_F67E36ECJ	BB_ITSYSTEMJDO_COMPONENTS_J
BB_HACMPLOCALRENCES_8ADCB6D9J	BB_SGROUPJDO_SENCES_CF68C060J
BB_HACMPLOCALREOLES_3C1CF07FJ	BB_SGROUPJDO_ROLES_J
BB_HACMPLOCALREOUPS_354DD1CCJ	BB_SERVICEGROUPOUPS_82AA2EE3J
BB_HACMPLOCALREROUP_21D3AC87J	BB_SERVICEGROUPOUPS_76D12CEDJ
BB_HACMPNODEJDONCES_F0DF78FDJ	BB_SFUNCTIONJDONCES_F8394E61J
BB_HACMPNODEJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_HIRDBSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_HIRDBSYSTEMJNCES_8CDFFAEEJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_IDSDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_IDSDATABASEJNCES_CFB10C79J	BB_SDEPLOYABLECNCES_572F3E83J
BB_IPNETWORKJDONCES_E3F0C245J	BB_SGROUPJDO_SENCES_CF68C060J
BB_IPNETWORKJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_ITSYSTEMJDO_NCES_31C6E3D2J	BB_SGROUPJDO_SENCES_CF68C060J
BB_ITSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_J2EEDOMAINJDNCES_111AC7A0J	BB_SGROUPJDO_SENCES_CF68C060J
BB_J2EEDOMAINJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_LINKSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_LINKSERVICEJNCES_F441B071J	BB_SDEPLOYABLECNCES_572F3E83J

Tableau 41. Les vues dépréciées et leurs nouveaux équivalents. (suite)

Vue obsolète	Nouvel équivalent
BB_LOGICALCONTENCES_F200987CJ	BB_SPHYSICALFILNCES_ECF04BA6J
BB_LOGICALCONTENTJDO_ROLES_J	BB_SPHYSICALFILEJDO_ROLES_J
BB_MBEXECUTIONGNCES_A5DA5D10J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBEXECUTIONGOLES_6D4906A8J	BB_SGROUPJDO_ROLES_J
BB_MBMESSAGEFLONCES_2C1D04EAJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBMESSAGEFLONCES_EB919DCCJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MBMESSAGEFLOOLES_18E0C30EJ	BB_SGROUPJDO_ROLES_J
BB_MBMESSAGEFLOWJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_MQINSTALLABLNCES_EDA0F908J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MQINSTALLABLOLES_688C35B0J	BB_SGROUPJDO_ROLES_J
BB_MQQUEUEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_MQQUEUEJDO_SNCES_CAE5631FJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_MSCLUSTERJDONCES_61127DF8J	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_MSCLUSTERNODEJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_MSCLUSTERNODNCES_3899EF16J	BB_SFUNCTIONJDONCES_F8394E61J
BB_MSCLUSTERRESNCES_22175CCFJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERRESNCES_A728F28AJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_MSCLUSTERRESOLES_324E6849J	BB_SGROUPJDO_ROLES_J
BB_MSCLUSTERRESOLES_3A01196EJ	BB_SGROUPJDO_ROLES_J
BB_NETWORKSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_NETWORKSERVINCES_CDD7C865J	BB_SDEPLOYABLECNCES_572F3E83J
BB_ORACLEDATABANCES_98E1A333J	BB_SDEPLOYABLECNCES_572F3E83J
BB_ORACLEDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_ORACLELISTENERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_ORACLELISTENNCES_72725D9AJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_ORACLESERVERJDO_ROLES_J	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_ORACLESERVERNCES_6F134AEBJ	BB_SSOFTWAREINSNCES_549D08D8J
BB_REALSERVERGRNCES_91B57D2EJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_REALSERVERGROUPJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_REALSERVERJDNCES_8B01D2CBJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_REALSERVERJDO_ROLES_J	BB_SSOFTWARESERVERJDO_ROLES_J
BB_SAMETIMESERVERJDO_ROLES_J	BB_SFUNCTIONJDO_ROLES_J
BB_SAMETIMESERVNCES_7FDA5716J	BB_SFUNCTIONJDONCES_F8394E61J
BB_SEGMENTJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_SEGMENTJDO_SNCES_CBC65999J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SERVICEGROUPROUP_6F30099EJ	BB_SERVICEGROUPOUPS_76D12CEDJ
BB_SERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SERVICEJDO_SNCES_63DE5757J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SMSHIERARCHYJDO_ROLES_J	BB_SGROUPJDO_ROLES_J

Tableau 41. Les vues dépréciées et leurs nouveaux équivalents. (suite)

Vue obsolète	Nouvel équivalent
BB_SMSHIERARCHYNCES_C6234B50J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SMSSITECOMPONCES_FC696136J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SMSSITECOMPOOLES_37331E42J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWARECOMPNCES_5E176596J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SOFTWARECOMPOLES_AA1C15E2J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWAREINSTNCES_EECCCFCBJ	BB_SSOFTWAREINSNCES_549D08D8J
BB_SOFTWAREINSTOLES_B502FACDJ	BB_SSOFTWAREINSOLES_7C7BE7E0J
BB_SOFTWAREMODULEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SOFTWAREMODUNCES_1FEAE999J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SPECIALITYSENCES_A28738B4J	BB_SFUNCTIONJDONCES_F8394E61J
BB_SPECIALITYSEOLES_33910984J	BB_SFUNCTIONJDO_ROLES_J
BB_SQLSERVERDATNCES_4C800F20J	BB_SDEPLOYABLECNCES_572F3E83J
BB_SQLSERVERDATOLES_E33DFE98J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_STORAGEEXTENNCES_614C0287J	BB_SDEPLOYABLECNCES_572F3E83J
BB_STORAGEEXTENTJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_STORAGEPOOLJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_STORAGEPOOLJNCES_AC507A15J	BB_SGROUPJDO_SENCES_CF68C060J
BB_SYBASEDATABANCES_DFF7B51AJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_SYBASEDATABASEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_SYSPLEXJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_SYSPLEXJDO_SNCES_4B3FE470J	BB_SGROUPJDO_SENCES_CF68C060J
BB_VCSSYSTEMJDONCES_831828D7J	BB_SGROUPJDO_SENCES_CF68C060J
BB_VCSSYSTEMJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBLOGICMACHINEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBLOGICMACHNCES_41F6228FJ	BB_SGROUPJDO_SENCES_CF68C060J
BB_WEBLOGICNODENCES_1032390BJ	BB_SSOFTWARESERNCES_19E863EFJ
BB_WEBLOGICNODEOLES_14E2D98DJ	BB_SSOFTWARESERVERJDO_ROLES_J
BB_WEBSPHERENODEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_WEBSPHERENODNCES_182E84E9J	BB_SGROUPJDO_SENCES_CF68C060J
BB_WINDOWSSERVICEJDO_ROLES_J	BB_SDEPLOYABLECOLES_C2D28F15J
BB_WINDOWSSERVINCES_9F30EF3AJ	BB_SDEPLOYABLECNCES_572F3E83J
BB_ZONEJDO_ROLES_J	BB_SGROUPJDO_ROLES_J
BB_ZONEJDO_SERVICEINSTANCES_J	BB_SGROUPJDO_SENCES_CF68C060J

Vues de la sous-fenêtre Détails

Vous pouvez utiliser le fichier `detail_panel_views.txt` pour écrire des requêtes et extraire des informations supplémentaires. Ces vues vous seront d'autant plus utiles que vous connaissez bien la sous-fenêtre Détails du portail de gestion de données.

Pour afficher la liste des vues disponibles de la sous-fenêtre Détails, accédez au répertoire `$COLLATION_HOME/etc/views` et ouvrez le fichier `detail_panel_views.txt`. Ce dernier contient la liste de toutes les sous-fenêtres Détails disponibles dans le portail de gestion de données, avec les vues de base de données correspondantes. Le nom de chaque vue de la sous-fenêtre Détails se présente sous la forme suivante :

```
DP_%_V
```

Dans chaque nom de vue, le symbole `%` représente le nom de la sous-fenêtre Détails dans le portail de gestion de données. Il existe plus de 600 vues de la sous-fenêtre Détails.

Des informations supplémentaires sur ces vues sont disponibles sous forme de commentaires dans les fichiers suivants se trouvant dans le même répertoire :

- Pour les bases de données DB2 : create_detail_panel_views_db2.sql
- Pour les bases de données Oracle : create_detail_panel_views_oracle.sql

Définitions de vue

Les vues de la sous-fenêtre Détails sont organisées en fonction des types d'objet du Common Data Model (CDM) des éléments de configuration affichés dans cette sous-fenêtre. Pour trouver les vues correspondant à un élément de configuration donné, cliquez sur l'onglet **Général** de la sous-fenêtre Détails et recherchez la valeur de la zone **Type d'objet**. Vous pouvez ensuite rechercher le type d'objet dans le fichier detail_panel_views.txt, qui identifie les vues correspondantes.

Par exemple, le fichier detail_panel_views.txt inclut l'entrée suivante pour le type d'objet `DB2 Instance` :

```
########## DB2 Instance...<Layout> ##########
...General.....................................................<Tab Level 1>
......General.....................................................<TabData>
......Db2Instance.General.....................................<Content>
.........DP_DB2_INSTANCE_GENERAL_V.............................<View>
...System......................................................<Tab Level 1>
......System Info.................................................<TabData>
......Db2Instance.SystemInfo..................................<Content>
.........DP_DB2_INSTANCE_SYSTEM_INFO_V.........................<View>
......Profile Registry............................................<TabData>
......Db2Instance.GlobalProfileRegistry.......................<Content>
.........DP_DB2_INSTANCE_GLOB_PROFREG_V.......................<View>
......License Info................................................<TabData>
......Db2Instance.LicenseInfo.................................<Content>
.........DP_DB2_INSTANCE_LICENSE_INFO_V.......................<View>
...Configuration...............................................<Tab Level 1>
......Configuration...............................................<TabData>
......Db2Instance.Configuration...............................<Content>
.........DP_DB2_INSTANCE_CONFIG_V.............................<View>
...Profile Registry............................................<Tab Level 1>
......Profile Registry............................................<TabData>
......Db2Instance.ProfileRegistry.............................<Content>
.........DP_DB2_INSTANCE_PROFILE_REG_V.......................<View>
...Databases...................................................<Tab Level 1>
......Databases...................................................<TabData>
......Db2Instance.Databases...................................<Content>
.........DP_DB2_INSTANCE_DATABASES_V..........................<View>
...Remote Databases............................................<Tab Level 1>
......Remote Databases............................................<TabData>
......Db2Instance.Db2Alias....................................<Content>
.........DP_DB2_INSTANCE_DB2_ALIAS_V..........................<View>
...Modules.....................................................<Tab Level 1>
......Software Modules............................................<TabData>
......AppServer.SoftwareModules..............................<Content>
.........DP_APP_SERVER_SW_MODULES_V...........................<View>
......Other Modules...............................................<TabData>
......AppServer.StaticModules................................<Content>
.........DP_APP_SERVER_STATIC_MOD_V...........................<View>
...Application Descriptors.....................................<Tab Level 1>
......Application Descriptors.....................................<TabData>
......AppServer.ApplicationDescriptors.......................<Content>
.........DP_APP_SERVER_APP_DESCRIPS_V.........................<View>
...Runtime.....................................................<Tab Level 1>
......Databases...................................................<TabData>
......Db2Instance.Runtime.....................................<Content>
.........DP_DB2_INSTANCE_RUNTIME_V...........................<View>
```

Cet exemple indique que l'onglet Général du type d'objet `DB2 instance` correspond à la vue de base de données DP_DB2_INSTANCE_GENERAL_V. Il affiche en outre la liste des vues de base de données supplémentaires disponibles pour les autres onglets (dont certains sont représentés par plusieurs vues).

Vous pouvez lancer une requête sur ces vues pour extraire des informations sur les éléments de configuration reconnus par TADDM. Par exemple, vous pouvez extraire des données de l'onglet Bases de données de `DB2 Instance` en lançant une requête sur la vue de base de données DP_DB2_INSTANCE_DATABASES_V. La sélection de chaque élément à partir de cette vue renvoie le contenu de l'onglet Base de données pour toutes les instances DB2 que TADDM a reconnues. Vous pouvez limiter la requête à une seule instance DB2 en associant la vue DP_DB2_INSTANCE_DATABASES_V à la vue DP_DB2_INSTANCE_GENERAL_V et en effectuant un filtrage par nom d'instance (voir «Exemple de requête», à la page 179).

Pour plus d'informations sur la façon dont ces vues sont définies, étudiez les commentaires du fichier create_detail_panel_views_db2.sql ou create_detail_panel_views_oracle.sql. Le bloc de commentaires suivant fournit des informations sur les noeuds, ou types d'objet CDM, référencés par la vue DP_DB2_INSTANCE_DATABASES_V :

```
-- ######## Db2Instance.Databases ########
--
-- View...... DP_DB2_INSTANCE_DATABASES_V
--
-- Node...... 1
-- Node Path..................... Db2Instance
-- Node Class Name............... model.topology.app.db.db2.Db2Instance
-- Node Type..................... Racine
-- Node...... 2
-- Node Path..................... Db2Instance._arraydatabases
-- Node Class Name............... model.topology.app.db.db2.Db2Database
-- Node Field Name............... databases
-- Node Type..................... Array Many-to-Many
-- Node...... 3
-- Node Path..................... Db2Instance._arraydatabases.databases
-- Node Class Name............... model.topology.app.db.db2.Db2Database
-- Node Field Name............... databases
-- Node Type..................... Array
```

Noeuds et colonnes

Pour savoir la façon dont les colonnes de vue sont définies, vous pouvez lancer une requête SQL `describe`. Par exemple, la requête suivante affiche les colonnes de la vue DP_DB2_INSTANCE_DATABASES_V :

```
db2 "describe table DP_DB2_INSTANCE_DATABASES_V"

Column                         Type      Type
name                           schema    name                Length   Scale Nulls
------------------------------ --------- ------------------ -------- ----- ------
NAME_C3                        SYSIBM    VARCHAR                 192     0 Yes
ALIAS_C3                       SYSIBM    VARCHAR                 192     0 Yes
PK_C3                          SYSIBM    VARCHAR                 192     0 Yes
PK_C1                          SYSIBM    VARCHAR                 192     0 No
```

Chaque nom de colonne se termine par un chiffre identifiant le noeud que représente la colonne. Cet exemple indique que les colonnes NAME_C3, ALIAS_C3 et PK_C3 font toutes référence au type d'objet `Db2Database` et que la colonne PK_C1 fait référence au type d'objet `Db2Instance`.

Remarque : Dans cet exemple, aucune colonne ne fait référence au noeud 2, car le tableau référencé est le tableau de mappage plusieurs-à-plusieurs connectant des instances DB2Instances à des bases de données DB2Databases. Le panneau Détails fait abstraction des complexités de l'association de ces deux tableaux, ce qui vous évite d'avoir à déterminer où les données sont stockées dans les tableaux de la base.

Exemple de requête

Avec toutes ces informations, vous pouvez afficher la liste des bases de données DB2 définies pour une instance particulière en utilisant la requête suivante :

```
select
   db.name_c3
from
   DP_DB2_INSTANCE_GENERAL_V inst
   join DP_DB2_INSTANCE_DATABASES_V db ON (inst.PK_C1 = db.PK_C1)
où :
   inst.db2_instance_c1 = 'bg-linux.tivlab.austin.ibm.com:db2inst1'
```

Cette requête associe les vues correspondant aux onglets Général et Bases de données, en utilisant la colonne PK_C1 (qui représente la clé principale de l'instance DB2) et en sélectionnant les noms de la base de données. Les résultats sont ensuite filtrés de façon à n'inclure que l'instance `bg-linux.tivlab.austin.ibm.com:db2inst1`.

Vues personnalisées

Des vues personnalisées sont fournies pour extraire des données de la base de données TADDM et pour aider à la création de rapports. TADDM fournit deux vues personnalisées qui sont définies dans le fichier `custom-views.xml`. Vous pouvez également définir des vues appelées vues définies par l'utilisateur, si les vues de bloc de construction et les vues de panneau Détails existantes ne vous donnent pas satisfaction.

Une vue personnalisée se définit à l'aide du langage XML et est ensuite traitée par des scripts TADDM afin de générer les instructions SQL `CREATE VIEW` requises. Une vue personnalisée est générée à partir des vues de bloc de construction existantes mais les scripts TADDM déterminent comment les associer.

TADDM contient la vue personnalisée CM_COMPUTER_SYSTEMS_V qui est décrite dans les exemples suivants. Cette vue contient les informations de base susceptibles d'être utilisées dans un rapport sur les systèmes informatiques reconnus par TADDM :

- Nom d'hôte complet
- Fabricant, modèle, numéro de série et type de châssis
- Type, nombre et vitesse des unités centrales
- Taille de RAM
- Système d'exploitation
- Adresses IP de tous les adaptateurs
- Capacité de stockage totale et espace disponible pour tous les systèmes de fichiers

Ces attributs proviennent de nombreux types d'objet Common Data Model (CDM) :

- Système informatique
- OperatingSystem
- Interface IP
- IPAddress
- Système de fichiers

En outre, les données contiennent deux relations du type "plusieurs-à-plusieurs" :

- ComputerSystem vers IPInterface
- ComputerSystem vers FileSystem

Pour écrire manuellement une requête sur ces informations, vous devez d'abord identifier toutes les vues de bloc de construction représentant les types et relations CDM. Vous devez ensuite déterminer la manière de les lier. En définissant une vue

utilisateur, vous pouvez utiliser les scripts TADDM pour définir automatiquement les liaisons requises et générer la requête SQL afin de créer les vues dont vous avez besoin.

Langage XML des vues définies par l'utilisateur

Les vues définies par l'utilisateur telles que les vues personnalisées sont créées à partir d'une définition XML. Pour créer une vue définie par l'utilisateur, effectuez les étapes suivantes :

1. Copiez le fichier `custom-views.xml` du répertoire `$COLLATION_HOME/etc/views` dans le répertoire `$COLLATION_HOME/bin`.
2. Renommez le fichier `custom-views.xml` en `user-views.xml`.
3. Modifiez les noms de vue CM_COMPUTER_SYSTEMS_V et CM_APP_SERVERS_PER_HOST_V dans le fichier `user-views.xml`. Ces vues sont réservées par TADDM et ne doivent pas être écrasées. Modifiez le reste du fichier selon vos besoins.

Elément	Attributs	Eléments contenus
`view`	**`className`** Désigne le nom de classe de l'objet de modèle de base de la vue. **`viewName`** Désigne le nom de la vue. Le nom doit être une chaîne commençant par `CM_`, finissant par `_V` et ne dépassant pas 30 caractères. Evitez les noms déjà utilisés. **`includePrimaryKeys`** Indique si les clés principales sont incluses sous forme de colonnes. La valeur doit être `true` ou `false`. Spécifiez `true` si la vue doit être associée à d'autres vues.	`field`
`field`	Néant	`nested` `plain`
`nested`	**`className`** Désigne le nom de classe de l'objet de modèle de base des éléments imbriqués. **`fieldName`** Désigne le nom de zone des éléments imbriqués.	`nested` `plain`

Elément	Attributs	Eléments contenus
`plain`	**`fieldName`** Désigne le nom de zone des éléments simples. **`nameInView`** Désigne le nom de la colonne à exposer dans la vue de la base de données. La longueur maximale est de 30 caractères. Evitez les mots réservés DB2 ou Oracle. **`displayType`** Désigne le type de valeur. Cet attribut est facultatif. Spécifiez une des valeurs suivantes : **`speed`** Une valeur en MHz **`memory`** Une valeur en Ko, Mo ou Go **`mBytes`** Une valeur en Mo **`date`** Un horodatage au format `AAAA-MM-JJ-HH24:MI:SS`, utilisé pour les zones contenant un temps en millisecondes **`networkSpeed`** Une valeur en mégabits par seconde **`StorageGBytes`** Une valeur en Go	Néant

Le langage XML décrit deux types de zones, toutes les deux contenues dans l'élément `field` :

- Une zone *plain* représente un attribut d'un objet de modèle. Par exemple, la zone "plain" suivante indique que l'attribut du nom de domaine complet (fully qualified domain name, FQDN) de `ComputerSystem` s'affiche dans la vue sous forme de colonne `FQDN` :

  ```
  <plain fieldName="fqdn" nameInView="FDQN"/>
  ```

- Une zone *nested* représente une relation entre des objets de modèle. Par exemple, la zone "nested" suivante décrit la relation entre `ComputerSystem` et `OperatingSystem` via l'attribut `OSRunning` du type d'objet `ComputerSystem` :

  ```
  <field>
    <nested className="com.collation.platform.model.topology.sys.OperatingSystem"
    fieldName="OSRunning">
      <plain fieldName="OSName" nameInView="OS_NAME"/>
    </nested>
  </field>
  ```

  Dans la zone "nested", une zone "plain" indique que le nom du système d'exploitation doit apparaître dans la vue en tant que colonne `OS_NAME`.

La définition XML complète de l'exemple de vue CM_COMPUTER_SYSTEMS_V est la suivante :

```
<views>
  <!-- ComputerSystems with OS, Filesystems -->
  <view className="com.collation.platform.model.topology.sys.ComputerSystem"
  viewName="CM_COMPUTER_SYSTEMS_V" includePrimaryKeys="false">
    <field>
      <plain fieldName="fqdn" nameInView="FDQN"/>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.net.IpInterface"
      fieldName="ipInterfaces">
        <nested className="com.collation.platform.model.topology.net.IpAddress"
        fieldName="ipAddress">
          <plain fieldName="dotNotation" nameInView="IP_ADDRESS"/>
          <plain fieldName="stringNotation" nameInView="IP_ADDRESS_STRING"/>
        </nested>
      </nested>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.sys.OperatingSystem"
      fieldName="OSRunning">
        <plain fieldName="OSName" nameInView="OS_NAME"/>
      </nested>
    </field>
    <field>
      <plain fieldName="CPUType" nameInView="CPU_TYPE"/>
      <plain fieldName="numCPUs" nameInView="NUM_CPUS"/>
      <plain displayType="speed" fieldName="CPUSpeed" nameInView="CPU_SPEED"/>
      <plain displayType="memory" fieldName="memorySize" nameInView="MEMORY_SIZE"/>
      <plain fieldName="primaryMACAddress" nameInView="PRIMARY_MAC_ADDRESS"/>
      <plain fieldName="serialNumber" nameInView="SERIAL_NUMBER"/>
      <plain fieldName="manufacturer" nameInView="MANUFACTURER"/>
      <plain fieldName="model" nameInView="MODEL"/>
      <plain fieldName="type" nameInView="SYSTEM_TYPE"/>
    </field>
    <field>
      <nested className="com.collation.platform.model.topology.sys.FileSystem"
      fieldName="fileSystems">
        <plain displayType="mBytes" fieldName="capacity" nameInView="CAPACITY"/>
        <plain displayType="mBytes" fieldName="availableSpace"
        nameInView="AVAILABLE_SPACE"/>
        <plain fieldName="displayName" nameInView="FILE_SYSTEM"/>
      </nested>
    </field>
  </view>

  <!-- AppServers with host -->
  <view className="com.collation.platform.model.topology.app.AppServer"
  viewName="CM_APP_SERVERS_PER_HOST_V" includePrimaryKeys="false">
    <field>
      <nested className="com.collation.platform.model.topology.sys.ComputerSystem"
      fieldName="host">
        <plain fieldName="fqdn" nameInView="FQDN"/>
        <nested className="com.collation.platform.model.topology.net.IpInterface"
        fieldName="ipInterfaces">
          <nested className="com.collation.platform.model.topology.net.IpAddress"
          fieldName="ipAddress">
            <plain fieldName="dotNotation" nameInView="IP_ADDRESS"/>
            <plain fieldName="stringNotation" nameInView="IP_ADDRESS_STRING"/>
          </nested>
        </nested>
      </nested>
    </field>

    <field>
      <plain fieldName="jdoClassName" nameInView="CLASS_NAME"/>
      <plain fieldName="objectType" nameInView="TYPE"/>
      <plain fieldName="productName" nameInView="PRODUCT_NAME"/>
```

```
      <plain fieldName="productVersion" nameInView="PRODUCT_VERSION"/>
      <plain fieldName="keyName" nameInView="KEY_NAME"/>
      <plain fieldName="name" nameInView="NAME"/>
    </field>
  </view>
</views>
```

Ajout de vues utilisateur dans la base de données

Une fois que vous avez défini vos vues utilisateur dans le fichier `user-views.xml`, procédez comme suit pour créer les vues dans la base de données :

1. A l'invite de la commande, accédez au répertoire `$COLLATION_HOME/bin`.
2. Exécutez l'une des commandes suivantes pour créer les scripts SQL requis :
   - Systèmes UNIX et Linux :`user_views.sh scripts`
   - Systèmes Windows :`user_views scripts`

   Cette commande crée les fichiers suivants :
   - `create_custom_views_db2.sql`
   - `create_custom_views_oracle.sql`
   - `create_custom_views_db2zos.sql`
   - `drop_custom_views_db2.sql`
   - `drop_custom_views_oracle.sql`
   - `drop_custom_views_db2zos.sql`
3. Exécutez l'une des commandes suivantes pour créer les vues dans votre base de données :
   - Systèmes UNIX et Linux :`user_views.sh recreate`
   - Systèmes Windows :`user_views recreate`

   Cette commande exécute le script SQL approprié à votre type de base de données.

Une fois que ces commandes ont été exécutées, vos vues utilisateur sont disponibles pour l'exécution de requêtes dans la base de données. Sachez que toute requête SQL que vous implémentez doit fournir le filtrage nécessaire de données renvoyées par ces vues (par exemple, en utilisant la clause `WHERE` SQL).

Modification des vues utilisateur

Pour modifier des vues utilisateur existant déjà dans la base de données :

1. Modifiez le fichier `user-views.xml` pour apporter les modifications nécessaires.
2. Répétez le processus de création des vues à l'aide de la commande **`user_views`**. Cette commande génère automatiquement les scripts SQL corrects permettant d'abandonner puis de recréer des vues modifiées.

   Remarque : Si vous renommez une vue utilisateur, vous devez abandonner manuellement la vue portant le nom d'origine avant d'exécuter les commandes permettant de créer la vue sous un nouveau nom.

Suppression de vues utilisateur

Pour supprimer une vue utilisateur de la base de données, exécutez la commande SQL DROP appropriée à votre base de données. Les commandes DROP pour les vues utilisateur sont générées par la commande **`user_views`** dans les fichiers suivants :

- `drop_custom_views_db2.sql`
- `drop_custom_views_oracle.sql`

- drop_custom_views_db2zos.sql

Vues des attributs étendus

Cet outil génère des vues de base de données qui font référence aux données pour les attributs étendus.

Pour chaque objet modèle comportant des attributs étendus, l'outil crée un script SQL qui lorsqu'il est exécuté, crée deux vues des attributs étendus :

- EA_*modèle*_V qui comporte des colonnes correspondant aux attributs étendus. Chaque attribut étendu peut être associé à la vue de bloc fonctionnel correspondant, BB_*modèle*_V, à l'aide de la colonne PK_C.
- Fix Pack 1 BE_*modèle*_V qui comporte des colonnes correspondant aux attributs Common Data Model et aux attributs étendus. Lorsqu'un nom de colonne est en conflit, les colonnes des attributs étendus pourraient ne pas apparaître.

Tous les attributs étendus sur le même type d'objet modèle sont dans la même vue de base de données.

Les scripts dépendent des définitions en cours des attributs étendus. L'outil ne contrôle pas les attributs qui ont été créés et supprimés, même si ces attributs avec valeurs sont toujours affectés à certains objets. Si vous modifiez un attribut étendu, vous devez supprimer la vue existante avant d'utiliser l'outil d'affichage des attributs étendus pour créer un script SQL et une vue des attributs à jour.

Fix Pack 3

Modification du type de données

Dans les versions TADDM 7.3.0.2 et antérieures, le script extattr_views.sh crée des vues d'attributs étendus avec des colonnes de type CLOB. A partir de la version 7.3.0.3, les vues comportent des types de données spécifiques, comme VARCHAR ou SMALLINT. La tableau suivant présente les nouveaux types de colonnes dans les vues d'attributs étendus dans les bases de données DB2 et Oracle.

Tableau 42. Types de colonnes dans les vues d'attributs étendus dans les bases de données DB2 et Oracle

Type d'interface graphique	Type de colonne dans DB2	Type de colonne dans Oracle
Chaîne	VARCHAR(32000)	VARCHAR2(4000)
Caractère	VARCHAR(4)	VARCHAR2(4)
Virgule flottante à double précision	DOUBLE	NUMBER
Virgule flottante	REAL	NUMBER
Booléen	SMALLINT	NUMBER(1)
Entier	INTEGER	NUMBER
Entier court	SMALLINT	NUMBER
Entier long	BIGINT	NUMBER

Une telle modification peut affecter l'intégration à des produits qui utilisent les vues de la base de données TADDM pour les attributs étendus. Par exemple, si

vous utilisez les rapports BIRT, il se peut que des erreurs soient générées si les colonnes des attributs étendus ne sont pas définies sur un type de données spécifique, par exemple, VARCHAR.

Remarque : Dans les versions TADDM 7.3.0.2 et antérieures, les attributs étendus de type booléen sont affichés en tant que `false` ou `true` dans les vues de base de données. Dans les versions TADDM 7.3.0.3 et ultérieures, la valeur est un entier correspondant à `0` ou 1. Ainsi, les rapports BIRT et Cognos créés avant l'application de cette modification ne sont pas compatibles avec les nouveaux types de données.

Syntaxe de la commande extattr_views.sh

Pour utiliser l'outil extattr_view, exécutez la commande `extattr_views.sh`, avec un paramètre de ligne de commande approprié. La commande `extattr_views.sh` se trouve dans le répertoire `$COLLATION_HOME/bin`.

Syntaxe de commande

extattr_views.sh *paramètre*

Paramètres

`scripts`
Permet de créer les scripts SQL suivants :

- `create_extattr_views_`*`type_bdd`*`.sql`
- `drop_extattr_views_`*`type_bdd`*`.sql`

où *type_bdd* est l'un des types de base de données suivants :

- db2
- oracle
- db2zos

`create`
Permet de créer des vues.

`remove`
Permet de supprimer des vues. Les scripts SQL correspondants ne sont pas supprimés.

Exécution de l'outil d'affichage des attributs étendus

Vous pouvez utiliser l'outil d'affichage des attributs étendus pour générer une vue de base de données qui correspond à un attribut étendu existant.

Procédure

Pour créer la vue correspondante d'un attribut étendu, procédez comme suit :

1. Créez un attribut étendu sur l'objet de modèle.
2. Utilisez l'outil d'affichage des attributs étendus pour créer les scripts SQL requis :
   ```
   extattr_views.sh scripts
   ```
3. Utilisez l'outil d'affichage des attributs étendus pour créer la vue :
   ```
   extattr_views.sh create
   ```
4. Facultatif : Interroger les données à l'aide d'une commande SQL.

Exemple

Par exemple, si vous créez un attribut étendu appelé 'SUPPORT_AREA' sur le type de modèle ComputerSystem, l'exécution de l'outil d'affichage des attributs étendus crée deux vues :

- EA_COMPUTERSYSTEM40_V
  La nouvelle vue possède les colonnes suivantes :
  - PK_C, qui contient la clé primaire
  - SUPPORT_AREA_C, qui contient les attributs étendus
- Fix Pack 1 BE_COMPUTERSYSTEM40_V
  La nouvelle vue possède les colonnes suivantes :
  - Toutes les colonnes issues de la vue de bloc fonctionnel BB_COMPUTERSYSTEM40_V, par exemple : PK_C, NAME_C, VMID_C.
  - SUPPORT_AREA_C, qui contient les attributs étendus, uniquement s'il n'y a aucun conflit de nom de colonne avec les colonnes issues de la vue de bloc fonctionnel BB_COMPUTERSYSTEM40_V.

Vous pouvez utiliser les commandes SQL suivantes pour demander les données :

```
SELECT
      T1.FQDN_C,
      T2.SUPPORT_AREA_C
     FROM
        BB_COMPUTERSYSTEM40_V T1,
        EA_COMPUTERSYSTEM40_V T2
     WHERE
        T1.PK_C = T2.PK_C
```

Fix Pack 1

```
SELECT
     FQDN_C, SUPPORT_AREA_C
    FROM
     BE_COMPUTERSYSTEM40_V
```

Remarque : Fix Pack 1 En cas de conflit sur un nom de colonne, l'outil tente d'ajouter la chaîne de catégorie des attributs étendus. Si cette opération ne résout pas le problème, la colonne des attributs étendus n'est pas incluse dans les vues BE_model_V.

Dictionnaire de données TADDM

Le dictionnaire de données TADDM est un ensemble de pages HTML générées automatiquement permettant de mapper les informations du Common Data Model à celles de la base de données TADDM.

Accès au dictionnaire de données

Le dictionnaire de données est disponible dans les emplacements suivants d'un serveur de stockage (dans un déploiement de serveur en continu), d'un serveur de synchronisation (dans un déploiement de serveur de synchronisation) ou d'un serveur de domaine (dans un déploiement de serveur de domaine) :

- A l'adresse http://*taddmserverhost*:*port*/cdm/datadictionary/, par exemple http://1.123.123.12:9430/cdm/datadictionary/

- Dans le fichier `taddm-data-dictionary.zip` qui se trouve dans les répertoires `$COLLATION_HOME/sdk/datadictionary` et `$COLLATION_HOME/deploy-tomcat/cdm/datadictionary` (TADDM 7.3.0) ou `$COLLATION_HOME/apps/cdm/datadictionary` (TADDM versions 7.3.0.1 et ultérieures).

  Pour utiliser le fichier `taddm-data-dictionary.zip`, effectuez les étapes suivantes :

  1. Extrayez le contenu du fichier `taddm-data-dictionary.zip` dans un emplacement de votre choix.
  2. Extrayez le fichier `$COLLATION_HOME/sdk/doc/model/CDMWebsite.zip` vers le répertoire `data-dictionary/cdm` de la structure du dictionnaire de données extraite.

Index

Le dictionnaire de données inclut les index suivants :

Index des vues de bloc de construction

Cet index est disponible à l'adresse suivante :

`http://`*`taddmserverhost`*`:`*`port`*`/cdm/datadictionary/bb-views/index.html`

Dans l'index, cliquez sur le nom d'une vue de bloc de construction. Les informations suivantes s'affichent :

- La table de base de données TADDM à partir de laquelle la vue de bloc de construction correspondante est remplie. Chaque nom de table renvoie à une définition de la table de base de données.
- Les colonnes de la vue de bloc de construction correspondante. Chaque nom de colonne renvoie à une définition CDM de l'attribut représenté par la colonne.

Index des objets de modèle

Cet index est disponible à l'adresse suivante :

`http://`*`taddmserverhost`*`:`*`port`*`/cdm/datadictionary/model-object/index.html`

Dans l'index, cliquez sur le nom d'une classe CDM. Les informations suivantes s'affichent :

- Les tables de la base de données TADDM contenant la classe CDM correspondante. Chaque nom de table renvoie à une définition de la table de base de données.
- Les vues des blocs de construction contenant la classe CDM correspondante. Chaque nom de vue de bloc de construction renvoie à une définition de vue de bloc de construction.

Index des tables d'objet de modèle

Cet index est disponible à l'adresse suivante :

`http://`*`taddmserverhost`*`:`*`port`*`/cdm/datadictionary/cdm-tables/index.html`

Dans l'index, cliquez sur le nom d'une table de base de données TADDM. Les informations suivantes s'affichent :

- La classe CDM qui déclare la table de base de données correspondante. Le nom de classe CDM renvoie à une définition de la classe.
- Les classes CDM contenues dans la base de données correspondante. Chaque nom de classe CDM renvoie à une définition de la classe.

- Les colonnes de la table de base de données correspondante. Chaque nom de colonne renvoie à une définition CDM de l'attribut représenté par la colonne.

Index des données reconnues par les détecteurs

Cet index est disponible à l'adresse suivante :

`http://taddmserverhost:port/cdm/datadictionary/sensors/index.html`

Dans l'index, cliquez sur le nom d'un détecteur. Les informations suivantes s'affichent :

- Les informations générales sur le détecteur.
- La classe CDM des objets de modèle reconnus par le détecteur. Chaque nom de classe CDM renvoie à une définition de la classe.
- Les attributs des classes CDM et les informations sur leur disponibilité dans le contexte du détecteur.

Index des données potentielles à reconnaître

Cet index est disponible à l'adresse suivante :

`http://taddmserverhost:port/cdm/datadictionary/sensors/potentialData.html`

Dans l'index, cliquez sur le nom d'une catégorie. Les noms des types de données qui appartiennent à la catégorie correspondante sont affichés.

Cliquez sur le nom d'un type de données. Les informations suivantes s'affichent :

- Les informations générales sur le détecteur qui reconnaît le type de données correspondant
- La classe CDM des objets de modèle reconnus par le détecteur. Chaque nom de classe CDM renvoie à une définition de la classe.
- Les attributs des classes CDM

Index supplémentaires

Dans le répertoire `datadictionary/cdm` du dictionnaire de données, se trouvent également les index suivants :

- Index de classe à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/classes/$index.htm`
- Index d'interface à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/interfaces/$index.htm`
- Index d'attribut à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/attributes/$index.htm`
- Index de relation à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/relationships/$index.htm`
- Index de type de données à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/datatypes/$index.htm`
- Index de règle de dénomination à l'adresse `http://taddmserverhost:port/cdm/datadictionary/cdm/namingrules/$index.htm`

Information JavadocTADDM

Vous pouvez utiliser l'information Javadoc inclue avec TADDM pour en savoir plus à propos des interfaces de programme d'application disponibles.

Les fichiers compressés suivants contiennent l'information Javadoc TADDM :

$COLLATION_HOME/sdk/doc/api/oalapi-javadoc.zip
Ce fichier contient des informations à propos de l'interface de programme d'application Java TADDM disponible.

$COLLATION_HOME/sdk/doc/api/taddmapi-javadoc.zip
Ce fichier contient des informations à propos de l'interface de programme d'application Java TADDM disponible.

$COLLATION_HOME/sdk/doc/capabilities/capabilities-javadoc.zip
Ce fichier contient des informations relatives à la fonctionnalité des fonctions que vous pouvez utiliser.

$COLLATION_HOME/sdk/doc/model/model-javadoc.zip
Ce fichier contient des informations à propos des objets du Common Data Model utilisés par TADDM.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est toutefois de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, programmes ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785 U.S.A.

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales :

LE PRESENT DOCUMENT EST LIVRE "EN L'ETAT" SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Certaines juridictions n'autorisent pas l'exclusion des garanties implicites, auquel cas l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, améliorer et/ou modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils

contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 Etats-Unis

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA (IBM Customer Agreement), des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Toute instruction relative aux intentions d'IBM pour ses opérations à venir est susceptible d'être modifiée ou annulée sans préavis, et doit être considérée uniquement comme un objectif.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

LICENCE DE COPYRIGHT :

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes

plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation IBM.

Toute copie totale ou partielle de ces programmes exemples et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

© (nom de votre société) (année). Des segments de code sont dérivés des Programmes exemples d'IBM Corp. © Copyright IBM Corp. _entrez l'année ou les années__. All rights reserved.

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur ne s'affichent pas.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et de services peuvent appartenir à IBM ou à des tiers. La liste actualisée de toutes les marques d'IBM est disponible sur la page Web «Copyright and trademark information» à l'adresse : http://www.ibm.com/legal/copytrade.shtml.

ITIL est une marque de The Office of Government Commerce et est enregistrée au bureau américain Patent and Trademark Office.

IT Infrastructure Library est une marque de The Central Computer and Telecommunications Agency, qui fait désormais partie de The Office of Government Commerce.

Java et toutes les marques et logos incluant Java sont des marques d'Oracle et/ou de ses filiales.

Linux est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Microsoft et Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

UNIX est une marque de The Open Group aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Imprimé en France