SIEMENS

Automatiser des projets à l'aide de scripts

Manuel système

Impression de l'aide en ligne

10/2016

Mentions légales

Signalétique d'avertissement

Ce manuel donne des consignes que vous devez respecter pour votre propre sécurité et pour éviter des dommages matériels. Les avertissements servant à votre sécurité personnelle sont accompagnés d'un triangle de danger, les avertissements concernant uniquement des dommages matériels sont dépourvus de ce triangle. Les avertissements sont représentés ci-après par ordre décroissant de niveau de risque.

⚠ DANGER
signifie que la non-application des mesures de sécurité appropriées **entraîne** la mort ou des blessures graves.

⚠ ATTENTION
signifie que la non-application des mesures de sécurité appropriées **peut entraîner** la mort ou des blessures graves.

⚠ PRUDENCE
signifie que la non-application des mesures de sécurité appropriées peut entraîner des blessures légères.

IMPORTANT
signifie que la non-application des mesures de sécurité appropriées peut entraîner un dommage matériel.

En présence de plusieurs niveaux de risque, c'est toujours l'avertissement correspondant au niveau le plus élevé qui est reproduit. Si un avertissement avec triangle de danger prévient des risques de dommages corporels, le même avertissement peut aussi contenir un avis de mise en garde contre des dommages matériels.

Personnes qualifiées

L'appareil/le système décrit dans cette documentation ne doit être manipulé que par du **personnel qualifié** pour chaque tâche spécifique. La documentation relative à cette tâche doit être observée, en particulier les consignes de sécurité et avertissements. Les personnes qualifiées sont, en raison de leur formation et de leur expérience, en mesure de reconnaître les risques liés au maniement de ce produit / système et de les éviter.

Utilisation des produits Siemens conforme à leur destination

Tenez compte des points suivants:

⚠ ATTENTION
Les produits Siemens ne doivent être utilisés que pour les cas d'application prévus dans le catalogue et dans la documentation technique correspondante. S'ils sont utilisés en liaison avec des produits et composants d'autres marques, ceux-ci doivent être recommandés ou agréés par Siemens. Le fonctionnement correct et sûr des produits suppose un transport, un entreposage, une mise en place, un montage, une mise en service, une utilisation et une maintenance dans les règles de l'art. Il faut respecter les conditions d'environnement admissibles ainsi que les indications dans les documentations afférentes.

Marques de fabrique

Toutes les désignations repérées par ® sont des marques déposées de Siemens AG. Les autres désignations dans ce document peuvent être des marques dont l'utilisation par des tiers à leurs propres fins peut enfreindre les droits de leurs propriétaires respectifs.

Exclusion de responsabilité

Nous avons vérifié la conformité du contenu du présent document avec le matériel et le logiciel qui y sont décrits. Ne pouvant toutefois exclure toute divergence, nous ne pouvons pas nous porter garants de la conformité intégrale. Si l'usage de ce manuel devait révéler des erreurs, nous en tiendrons compte et apporterons les corrections nécessaires dès la prochaine édition.

Siemens AG
Division Digital Factory
Postfach 48 48
90026 NÜRNBERG
ALLEMAGNE

℗ 10/2016 Sous réserve de modifications

Sommaire

Consignes de sécurité

1

Informations de sécurité

Siemens commercialise des produits et solutions comprenant des fonctions de sécurité industrielle qui contribuent à une exploitation sûre des installations, solutions, machines, équipements et/ou réseaux.

Pour protéger les installations, les systèmes, les machines et les réseaux des cybermenaces, il est nécessaire d'intégrer un concept de sécurité industrielle moderne complet et d'en assurer la maintenance. Les produits et solutions de Siemens ne constituent qu'une partie d'un tel concept.

Il incombe au client d'empêcher tout accès non autorisé à ses installations, systèmes, machines et réseaux. Les systèmes, machines et composants doivent uniquement être connectés au réseau d'entreprise ou à Internet dans la mesure où c'est nécessaire et en appliquant des mesures de protection correspondantes (p. ex. utilisation de pare-feux et segmentation du réseau).

En outre, il convient de respecter les directives de Siemens relatives aux mesures de sécurité correspondantes. Pour plus d'informations sur la sécurité industrielle, rendez-vous sur

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Les produits et solutions Siemens font l'objet de développements continus pour être plus sûrs. Siemens recommande d'utiliser impérativement les mises à jour des produits dès qu'elles sont disponibles ainsi que les versions de produits les plus récentes. L'utilisation de versions de produits qui ne sont plus supportées et le non-respect des mises à jour les plus récentes peut augmenter le risque d'exposition du client à des cybermenaces.

Afin d'être informé des mises à jour des produits, veuillez souscrire au flux RSS Siemens Industrial Security à l'adresse

http://www.siemens.com/industrialsecurity (http://www.industry.siemens.com/topics/global/en/industrial-security/Pages/Default.aspx)

Lisezmoi TIA Portal Openness 2

2.1 Lisezmoi

Copie d'Openness

Lorsque vous copiez une application Openness exécutable, il est possible que le chemin de répertoire qui est lu soit encore celui où l'application Openness a été créée à l'origine.

Solution :

Lorsque vous avez copié l'application Openness dans un nouveau répertoire, ouvrez par exemple la boîte de dialogue "Propriétés", puis refermez-la pour mettre à jour le cache de Windows.

Compatibilité multi-utilisateur d'Openness

Remarque

Openness ne prend pas en charge le mode multi-utilisateur, car l'utilisation d'Openness dans des projets multi-utilisateurs ne peut pas être recommandée. Notez que certaines actions Openness peuvent même perturber le bon déroulement des opérations multi-utilisateurs définies par l'interface utilisateur de TIA Portal.

Toutefois, si vous souhaitez effectuer des modifications avec Openness, exportez d'abord le projet multi-utilisateur dans un projet unique.

Amélioration du niveau de performance d'Openness

Pour atteindre le niveau de performance maximal d'Openness, vous pouvez désactiver la fonction de recherche globale de TIA Portal. Pour désactiver la recherche globale, utilisez l'interface utilisateur de TIA Portal, puisque Openness n'offre pas d'appel d'API à cet effet. À la fin du script Openness, vous pouvez de nouveau activer la recherche globale. La désactivation de la recherche globale permet certes d'améliorer le niveau de performance, mais toutes les fonctions Openness marchent normalement même lorsque la recherche globale est activée.

Failsafe et Openness

Remarque

Si vous utilisez Openness, vous devez tenir compte de certaines restrictions concernant la sécurité en cas de défaut. Pour plus d'informations, voir la documentation correspondante.

Code du programme à thread sécurisé

Remarque

Notez que votre code est à thread sécurisé, un événement apparaît dans différents threads.

Comportement d'exportation d'éléments d'écran avec style activé

Lors de l'exportation d'un élément d'écran avec style activé, les propriétés de l'élément de style ne sont pas exportées, mais uniquement les propriétés de l'élément d'écran avant l'activation du style. Lorsqu'un style est sélectionné et que "UseDesignColorSchema" est activé pour l'élément d'écran, l'élément d'écran prend les valeurs des propriétés du style sur l'interface utilisateur. Toutefois, dans la base de données, les propriétés définies avant la sélection du style restent enregistrées pour cet élément d'écran. Openness exporte les valeurs enregistrées dans la base de données.

Lorsque le style est désactivé puis réactivé et que l'élément d'écran est à nouveau exporté, ce sont les valeurs des propriété applicables dans l'élément de style qui sont exportées pour cet élément d'écran, car les valeurs des propriétés de l'élément de style sélectionné pour cet élément d'écran sont enregistrées dans la base de données lorsque "UseDesignColorSchema" n'est pas activé.

Ce problème peut être résolu de la manière suivante :

1. Affectez l'élément d'écran à l'élément de style :
   - la base de données contient les valeurs de propriétés valables avant l'activation du style.
   - L'interface utilisateur reprend les propriétés directement de l'élément de style.
2. Exportez l'élément d'écran affecté à l'élément de style :
   - Le fichier XML contient les valeurs des propriétés issues de la base de données, qui correspondent aux valeurs antérieures à l'activation du style.
3. Désactivez "UseDesignColorSchema" :
   - Les valeurs des propriétés de l'élément de style sont ajoutés aux propriétés de l'élément d'écran dans la base de données.
4. Activez "UseDesignColorSchema" :
   - Les valeurs des propriétés de l'élément d'écran ne sont pas modifiées dans la base de données et correspondent toujours à celles de l'étape 3.
   - L'interface utilisateur reprend les propriétés directement de l'élément de style.
5. Exportez l'élément d'écran affecté à l'élément de style :
   - Le fichier XML contient les valeurs des propriétés issues de la base de données, qui ont été définies à l'étape 3 et qui correspondent aux valeurs de l'élément de style.

Nouveautés d'Openness V14 3

Nouveau modèle d'objet Openness

Par comparaison à Openness V13 SP1, le modèle d'objet dans Openness V14 a changé.

Remarque

Le code de programme des applications, qui ont été utilisées pour Openness jusqu'à V13 SP1, doit être mis à jour.

Pour en savoir plus sur le nouveau modèle d'objet Openness, voir Modèle d'objet Openness V14 (Page 45) et Les modifications les plus importantes dans Openness V14 (Page 273).

Voir aussi

Traitement des exceptions (Page 193)

Principales modifications de chaîne de caractères (Page 276)

notions de base

4

4.1 Conditions requises pour TIA Portal Openness V14

Conditions requises pour l'utilisation d'applications Openness

- Un produit basé sur TIA Portal est installé sur le PC, tel que "STEP 7 Professional" ou "WinCC Professional".
- Le complément "TIA Portal Openness V14 " est installé sur le PC.
  Voir Installation de TIA Portal Openness V14 (Page 17)

Systèmes d'exploitation Windows pris en charge

Le tableau suivant montre les combinaisons du système d'exploitation Windows, de TIA Portal et de l'application utilisateur qui sont compatibles :

Système d'exploitation Windows	TIA Portal	Application utilisateur
64 bits	64 bits	32 bits, 64 bits et AnyCPU

Conditions requises pour la programmation d'applications Openness

- Microsoft Visual Studio 2015 Update 1 ou supérieur avec .Net 4.6.1.

Savoir-faire nécessaire de l'utilisateur

- Connaissances en ingénierie système
- Connaissances avancées de Microsoft Visual Studio 2015 Update 1 ou supérieur avec .Net 4.6.1
- Connaissances avancées de C# / VB.net et .Net
- Connaissances sur l'utilisation de TIA Portal

Canaux Openness Remoting

Les canaux Openness Remoting sont enregistrés comme type IpcChannel, le paramètre "ensureSecurity" étant mis sur "false".

Remarque

Evitez d'enregistrer un autre IpcChannel avec le paramètre "ensureSecurity" différent de "false" et une priorité supérieure ou égale à "1".

Les attributs suivants sont définis pour le IpcChannel :

Attribut	Paramètres
"name" et "portName"	Mis sur $"{Process.Name}_{Process.Id}" ou $"{Process.Name}_{Process.Id}_{AppDomain.Id}" si enregistré dans un autre AppDomain que l'AppDomain par défaut de l'application.
"priority"	Mis à la valeur par défaut "1".
"typeFilterLevel"	Mis sur 'Elevé".
"authorizedGroup"	Mis sur la chaîne de caractères de la valeur de compte NT pour le compte d'utilisateur intégré (c.-à-d. Tous)

Voir aussi

Ajouter un utilisateur au groupe d'utilisateurs "Siemens TIA Openness" (Page 19)

4.2 Installation

4.2.1 Installation de TIA Portal Openness V14

Introduction

Le complément "TIA Portal Openness V14 " est installé automatiquement par un programme d'installation.

Le fichier d'installation ""Siemens_TIA_Openness_V14.exe" est une archive à décompression automatique se trouvant sur le DVD du produit, dans le répertoire "Support"".

Conditions

- Le matériel et les logiciels de l'appareil de programmation ou du PC sont conformes à la configuration système requise.
- Vous détenez les droits d'administrateur.
- Les programmes en cours d'exécution ont été fermés.
- La fonction d'exécution automatique est désactivée.
- WinCC V14 et/ou STEP 7 V14 est/sont installé(s).
- Le numéro de version du complément "TIA Portal Openness" correspond aux numéros de version de WinCC et STEP 7.

Remarque

Public API - Openness V13 SP1 et Openness V14

Si vous installez Openness V14, Openness V14 est installé à côté d'Openness V13 SP1.

Marche à suivre

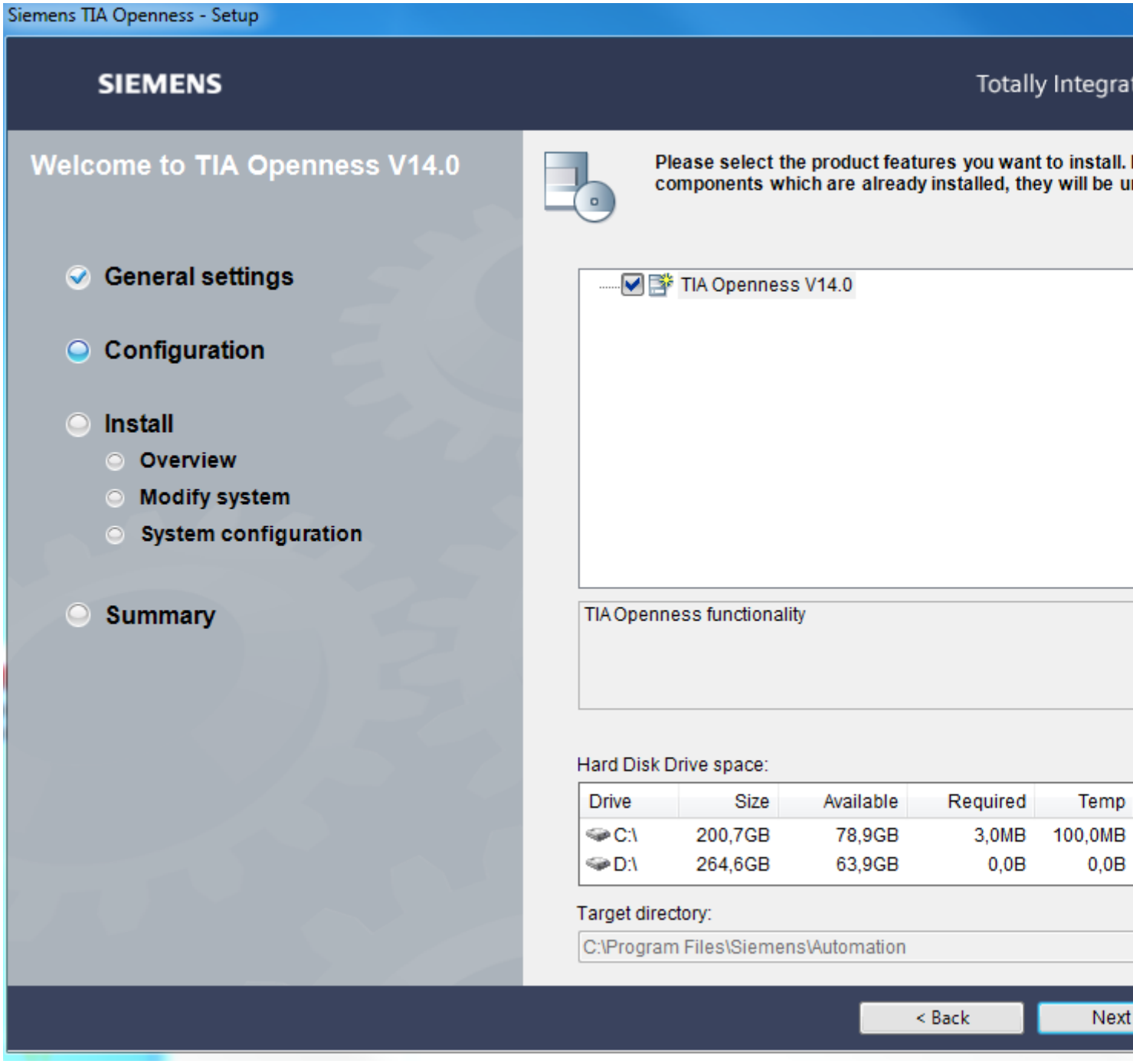
Remarque

Le complément "TIA Portal Openness V14 " ne peut être installé que lorsque l'installation de TIA Portal est terminée.

Pour installer le complément, procédez comme suit :

1. Insérez le support de données d'installation dans le lecteur correspondant et ouvrez répertoire "Support"
2. Double-cliquez sur le fichier "Siemens_TIA_Openness_V14.exe.
3. Sélectionnez la langue utilisée pour l'installation.

4. Les données d'installation sont extraites. Le programme d'installation d'Openness est lancé après l'extraction.
   La boîte de dialogue d'installation suivante apparaît :
5. Cliquez sur "Next" puis sélectionnez l'option requise.
6. Cliquez sur "Next" et sélectionnez le produit "TIA Portal Openness V14.0".



Au besoin, modifiez le répertoire cible pour l'installation et cliquez sur "Next".
Notez bien que le chemin d'installation ne doit pas dépasser 89 caractères.

7. A l'étape suivante de l'installation, acceptez tous les accords de licence puis cliquez sur "Next".
   Si l'installation de TIA Portal Openness nécessite une modification préalable des paramètres de sécurité et des droits d'accès, la boîte de dialogue des paramètres de sécurité s'ouvre.
8. Afin de poursuivre l'installation, confirmez les modifications des paramètres de sécurité et des droits puis cliquez sur "Next".
   La boîte de dialogue suivante présente les paramètres d'installation :
9. Vérifiez les paramètres d'installation sélectionnés et modifiez-les au besoin.
10. Cliquez sur Install pour démarrer l'installation.
    Si elle se déroule correctement, un message correspondant s'affiche à l'écran. Si des erreurs sont survenues pendant l'installation, un message d'erreur indiquant la cause de l'erreur s'affiche.

Résultat

Le complément "TIA Portal Openness V14 " est installé sur le PC. En outre, le groupe d'utilisateurs local "Siemens TIA Openness" est créé.

Remarque

Le complément "TIA Portal Openness V14 " ne suffit pas pour continuer d'avoir accès à TIA Portal. Vous devez être membre du groupe d'utilisateurs "Siemens TIA Openness" (voir Ajouter un utilisateur au groupe d'utilisateurs "Siemens TIA Openness" (Page 19)).

4.2.2 Ajouter un utilisateur au groupe d'utilisateurs "Siemens TIA Openness"

Introduction

Lorsque vous installez TIA Portal Openness V14 sur le PC, le groupe d'utilisateurs "Siemens TIA Openness" est automatiquement créé.

Chaque fois que vous accédez à TIA Portal avec votre application Openness, TIA Portal vérifie si vous êtes membre du groupe d'utilisateurs "Siemens TIA Openness", soit directement ou indirectement via un autre groupe d'utilisateurs. Si vous êtes membre du groupe d'utilisateurs "Siemens TIA Openness", l'application Openness démarre et établit une liaison à TIA Portal.

Marche à suivre

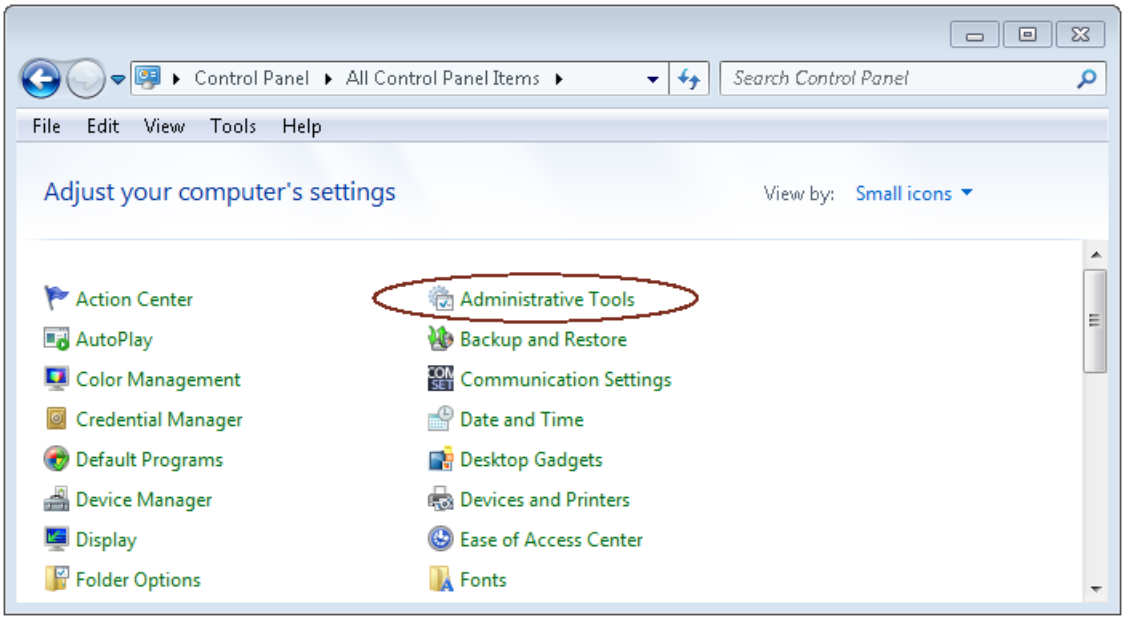
Vous ajoutez un utilisateur au groupe d'utilisateurs "Siemens TIA Openness" grâce à des applications de votre système d'exploitation. Le portail TIA ne prend pas en charge ce processus.

Remarque

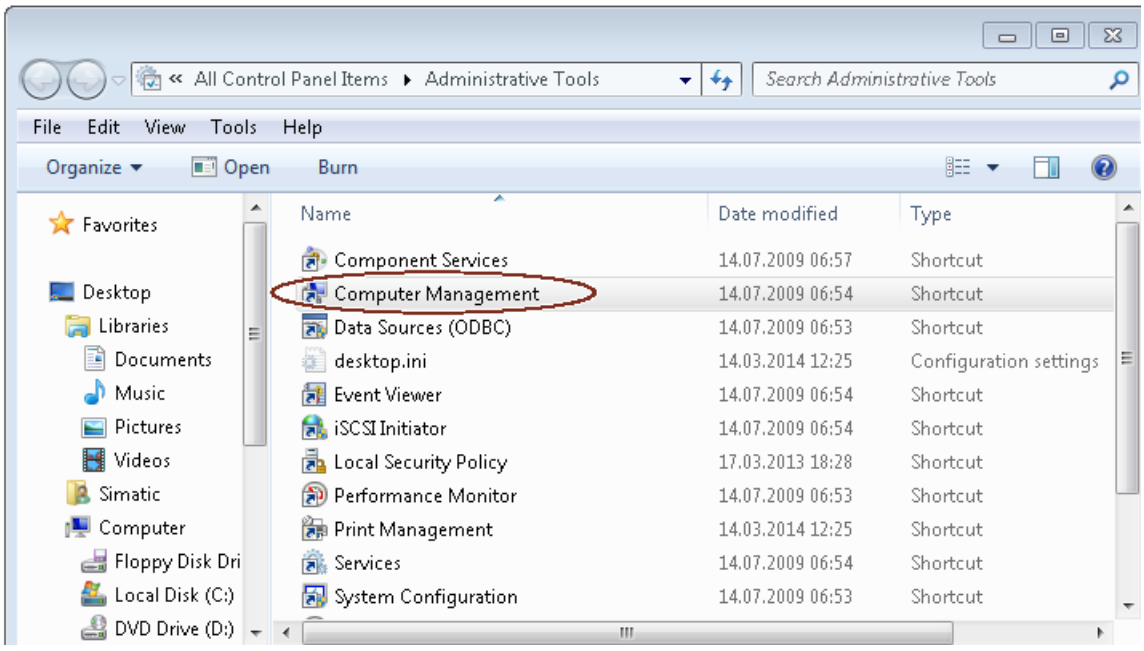
En fonction de la configuration de votre domaine ou de votre PC, les droits d'administrateur sont requis pour l'extension d'un groupe d'utilisateurs.

Avec le système d'exploitation Windows 7 (avec l'anglais comme langue paramétrée), vous pouvez par exemple ajouter un utilisateur au groupe d'utilisateurs comme suit :

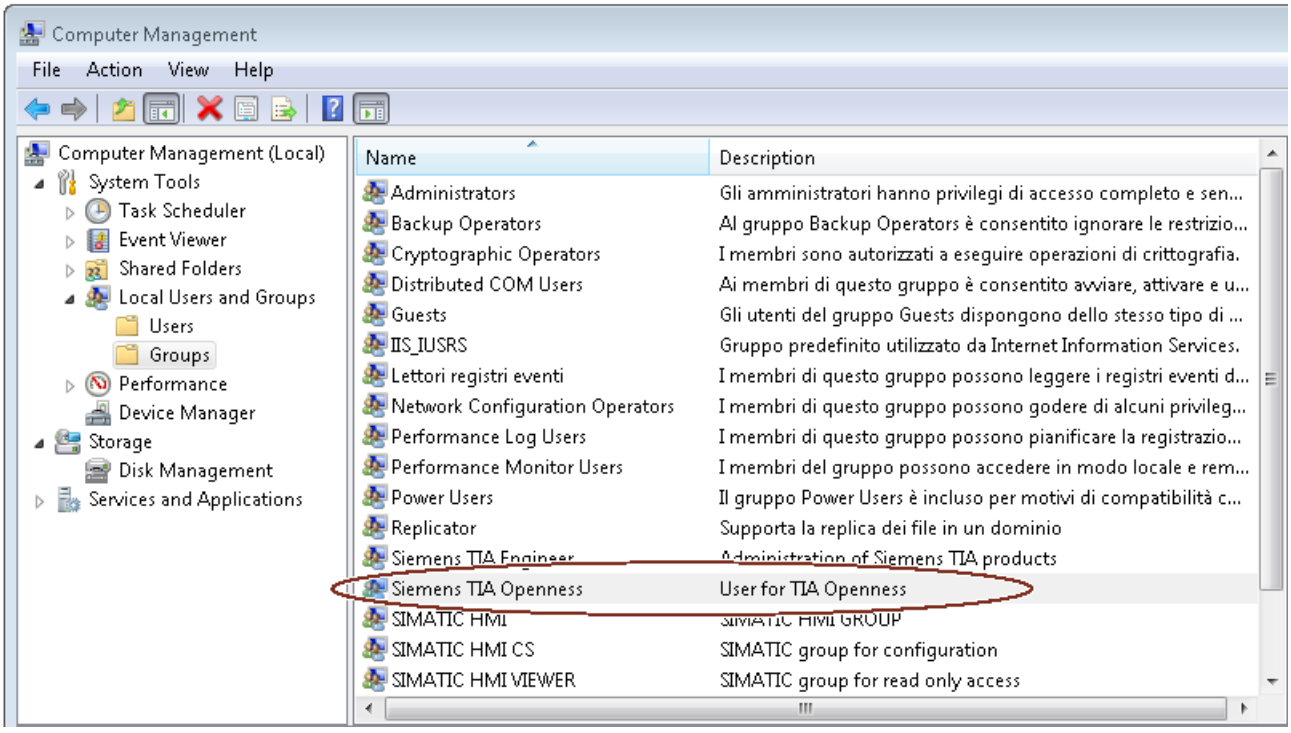
1. Sélectionnez "Start" > "Control Panel".
2. Double-cliquez dans le panneau de configuration sur "Administrative Tools".



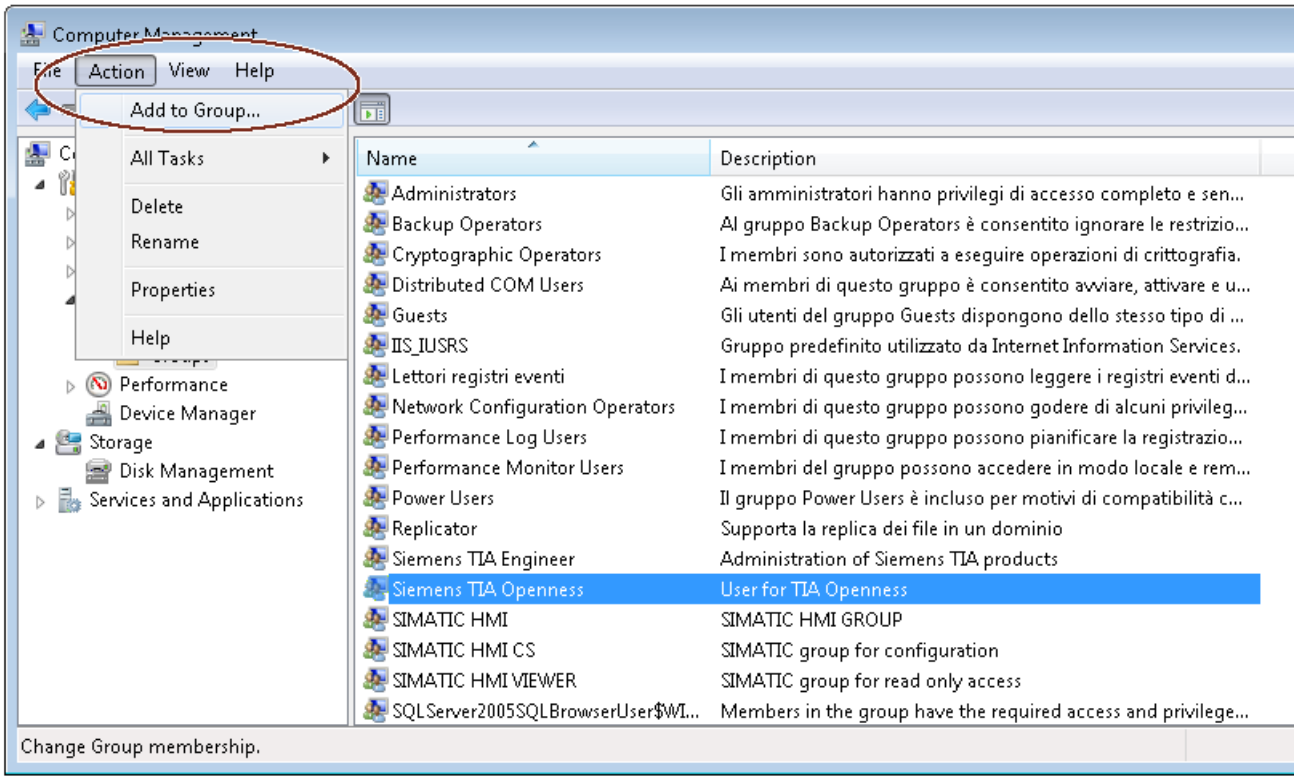
3. Cliquez sur "Computer Management" pour ouvrir la boîte de dialogue de configuration du même nom.



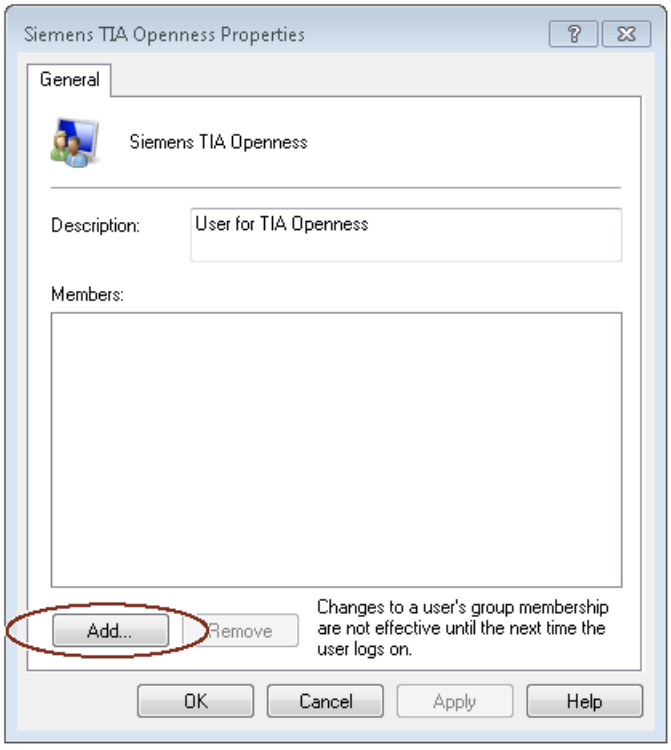
4. Sélectionnez "Local Users and Groups > Groups" pour afficher tous les groupes d'utilisateurs créés.
5. Dans la liste des groupes d'utilisateurs, sélectionnez dans le volet à droite l'entrée "Siemens TIA Openness".



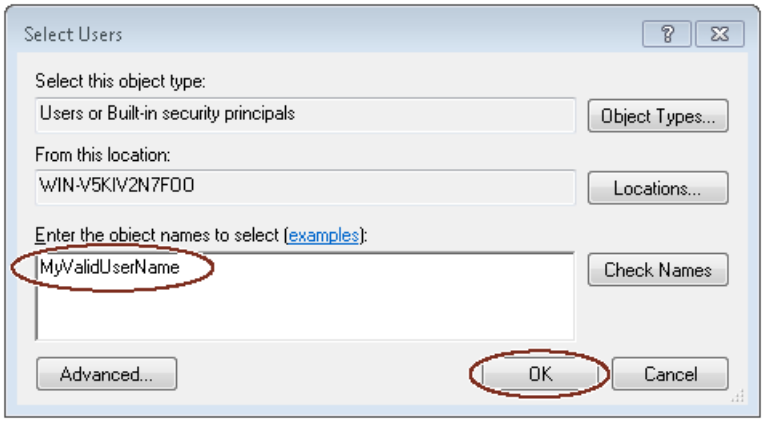
6. Sélectionnez la commande de menu "Action > Add to Group...".



La boîte de dialogue des propriétés du groupe d'utilisateurs s'ouvre :



7. Cliquez sur "Add".
   Une boîte de dialogue de sélection regroupant les utilisateurs pouvant être sélectionnés s'ouvre alors :



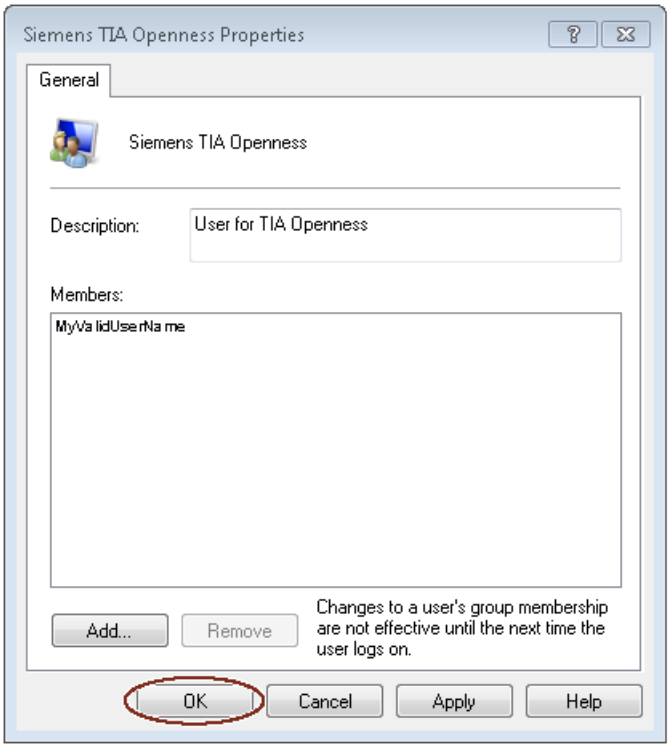
8. Entrez un nom d'utilisateur valide dans le champ de saisie.

Remarque

Cliquez sur le bouton "Check Names" pour vérifier si l'utilisateur saisi dispose d'un compte utilisateur valable pour ce domaine ou cet ordinateur.

Le champ "From this location" affiche le domaine ou le nom de l'ordinateur du nom d'utilisateur saisi. Vous pouvez obtenir des informations supplémentaires auprès de votre administrateur système.

9. Confirmez votre sélection par "OK".
   Le nouvel utilisateur s'affiche alors dans la boîte de dialogue des propriétés du groupe d'utilisateurs.

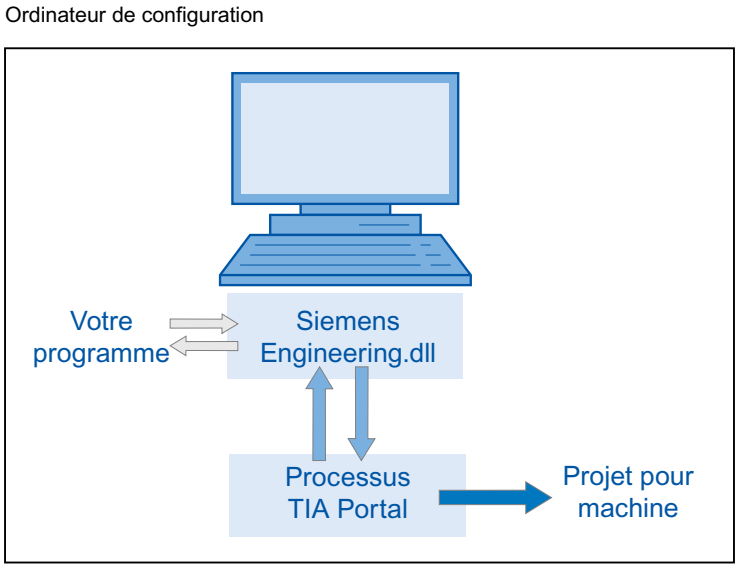


   Vous saisissez d'autres utilisateurs à l'aide du bouton "Add".

10. Cliquez sur "OK" pour mettre fin à cette opération.
11. Connectez-vous de nouveau au PC pour que les modifications deviennent effectives.

4.2.3 Accéder au portail TIA

Vue d'ensemble



Marche à suivre

1. Pour accéder et lancer TIA Portal, configurez votre environnement de développement.
2. Dans votre programme, créez une instance de l'objet de l'application de TIA Portal pour démarrer ce dernier.
3. Cherchez le projet souhaité et ouvrez-le.
4. Accédez aux données de projet.
5. Fermez le projet et quittez le portail TIA.

Voir aussi

Etablissement d'une connexion au portail TIA (Page 69)

Mettre fin à la connexion au portail TIA (Page 79)

4.2.4 Enabler File et Usage File

Introduction

Pour importer des blocs d'un appareil API, il vous faut deux fichiers :

- Enabler-File
- Usage-File

Vous trouverez la description indiquant comment vous procurer les deux fichiers dans l'autorisation de livraison STEP 7 V14, à la fin de l'article intitulé "Documents d'information".

Mot clé : TIA Portal Openness, instructions pour EnablerFile/UsageFile : TIA Portal Openness - Instructions Enabler File et Usage File (http://support.automation.siemens.com/WW/view/en/103627307)

Fonction des fichiers Enabler File et Usage File

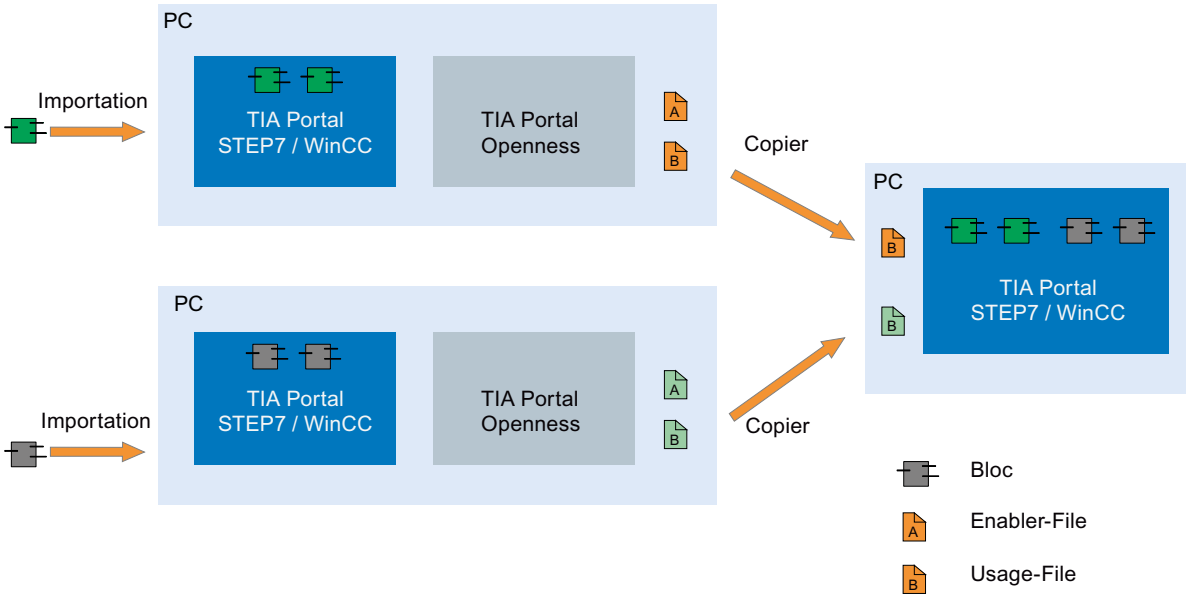
L'importation et l'utilisation de blocs d'un appareil API requièrent deux fichiers différents :

Pour importer des blocs, il vous faut les fichiers "Enabler-File" et "Usage-File". Pour pouvoir utiliser le bloc importé sur le même PC, il vous faut le fichier "Usage-File".

Si vous voulez copier et utiliser le bloc importé sur un autre PC, vous devez également copier le fichier "Usage-File" du bloc copié sur l'autre PC.

Remarque

Le fichier Enabler-File et le fichier d'exécution de votre application doivent se trouver dans le même répertoire. Le fichier Usage-File doit se trouver dans le répertoire PublicAPI/V14/ ou dans un sous-répertoire de celui-ci. Si vous ajoutez ces fichiers pendant l'exécution de TIA Portal, vous devez redémarrer TIA Portal pour pouvoir les utiliser. Évitez de renommer les fichiers Enabler-File ou Usage-File.



4.3 Tâches d'Openness

4.3.1 Possibilités d'utilisation

Introduction

TIA Portal Openness V14 vous offre différentes possibilités d'accès à TIA Portal et une sélection de fonctions pour des tâches définies.

Vous accédez aux zones suivantes de TIA Portal par le biais de l'interface Public API :

- Données du projet
- Données API
- Données IHM

Remarque

L'interface Public API ne doit pas servir à procéder à des contrôles ou à créer des données utilisées pour la réception/validation d'une installation de sécurité. Seuls une impression de sécurité réalisée avec le progiciel STEP 7 Safety ou le test fonctionnel conviennent pour une réception/valiation. Public API ne saurait les remplacer.

Accéder à TIA Portal

Avec TIA Portal Openness V14 , vous disposez de différentes possibilités d'accès à TIA Portal. Ce faisant, vous créez une instance externe de TIA Portal dans le processus, avec ou sans interface utilisateur. Vous pouvez également accéder en parallèle à des processus en cours de TIA Portal.

Accéder aux projets et données de projet

Lorsque vous accédez à des projets et des données de projet, vous utilisez TIA Portal Openness V14 principalement pour les tâches suivantes :

- Ouvrir, fermer et enregistrer un projet
- Enumérer et interroger des objets
- Créer des objets
- Supprimer des objets

4.3.2 Exportation/importation

Introduction

TIA Portal Openness V14 prend en charge l'importation et l'exportation de données de projet au moyen de fichiers XML. La fonction d'importation/exportation permet la configuration externe de données d'ingénierie existantes. Vous utilisez cette fonction pour rendre votre processus d'ingénierie plus efficace et éviter les erreurs.

Utilisation

Vous utilisez la fonction d'importation/exportation aux fins suivantes :

- Echange de données
- Copie de parties d'un projet
- Traitement externe de données de configuration, par exemple pour des opérations sur données de masse avec rechercher et remplacer
- Traitement externe de données de configuration pour de nouveaux projets sur la base de configurations existantes
- Importation de données de configuration générées en externe, telles que des listes de textes et des variables
- Mise à disposition de données de projet pour des applications externes

4.4 Liste d'objets

Introduction

Les tableaux suivants indiquent les objets disponibles, y compris Runtime Advanced et si ces objets sont pris en charge par Openness.

Ni le logiciel de visualisation Runtime Professional ni les fichiers de proxy d'appareil ne sont pris en charge par Openness dans WinCC V14.

Objets

Selon le pupitre opérateur que vous utilisez vous pouvez avoir recours aux données de projet suivantes :

Tableau 4-1 Vues

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Vue	oui	oui	oui	oui	oui	oui
Vue globale	oui	oui	oui	oui	oui	oui
Modèles	oui	oui	oui	oui	oui	oui
Fenêtre permanente	non	oui	oui	oui	oui	oui
Vue contextuelle	non	non	non	non	non	non
Vue coulissante	non	non	non	non	non	non

Tableau 4-2 Objets de vue

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Ligne	oui	oui	oui	oui	oui	oui
Ligne polygonale	non	oui	oui	oui	oui	oui
Polygone	non	oui	oui	oui	oui	oui
Ellipse	oui	oui	oui	oui	oui	oui
Segment d'ellipse	non	non	non	non	non	non
Segment de cercle	non	non	non	non	non	non
Arc d'ellipse	non	non	non	non	non	non
Affichage caméra	non	non	non	non	non	non
Arc de cercle	non	non	non	non	non	non
Cercle	oui	oui	oui	oui	oui	oui
Affichage PDF	non	non	non	non	non	non
Rectangle	oui	oui	oui	oui	oui	oui
Connecteur	non	non	non	non	non	non

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Champ de texte	oui	oui	oui	oui	oui	oui
Affichage graphique	oui	oui	oui	oui	oui	oui
Tuyau	non	non	non	non	non	non
Double raccord en T	non	non	non	non	non	non
Raccord en T	non	non	non	non	non	non
Coude	non	non	non	non	non	non
Champ d'E/S	oui	oui	oui	oui	oui	oui
Champ date/ heure	oui	oui	oui	oui	oui	oui
Champ d'E/S graphique	oui	oui	oui	oui	oui	oui
Champ de texte éditable	non	non	non	non	non	non
Champ de liste	non	non	non	non	non	non
Zone de liste déroulante	non	non	non	non	non	non
Bouton	oui	oui	oui	oui	oui	oui
Bouton rond	non	non	non	non	non	non
Bouton-poussoir lumineux	non	non	non	non	oui	non
Commutateur	oui	oui	oui	oui	oui	oui
Champ d'E/S symbolique	oui	oui	oui	oui	oui	oui
Commutateur à clé	non	non	non	non	oui	non
Bargraphe	oui	oui	oui	oui	oui	oui
Bibliothèque d'icônes	non	oui	oui	oui	oui	oui
Curseur	non	oui	oui	oui	oui	oui
Barre de défilement	non	non	non	non	non	non
Case à cocher	non	non	non	non	non	non
Bouton d'option	non	non	non	non	non	non
Instrument à aiguille	non	oui	oui	oui	oui	oui
Horloge	non	oui	oui	oui	oui	oui
Vue de l'espace mémoire	non	non	non	non	non	non
Touches de fonction	oui	oui	oui	oui	oui	oui
Instances de bloc d'affichage	non	non	non	non	non	non
Fenêtre de vues	non	non	non	non	non	non

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Vue des utilisateurs	oui	oui	oui	oui	oui	oui
Navigateur HTML	non	non	non	non	non	non
Travail d'impression/Diagnostic de script	non	non	non	non	non	non
Vue de recette	non	non	non	non	non	non
Vue des alarmes	non	non	non	non	non	non
Indicateur d'alarme	non	non	non	non	non	non
Fenêtre d'alarmes	non	non	non	non	non	non
Vue de courbes f(x)	non	non	non	non	non	non
Vue de courbes f(t)	non	non	non	non	non	non
Vue tabellaire	non	non	non	non	non	non
Table des valeurs	non	non	non	non	non	non
Media Player	non	non	non	non	non	non
Diagnostic de voie	non	non	non	non	non	non
WLAN - Réception	non	non	non	non	non	non
Zone - Nom	non	non	non	non	non	non
Zone - Signal	non	non	non	non	non	non
Nom de la plage d'action	non	non	non	non	non	non
Nom de la plage d'action (RFID)	non	non	non	non	non	non
Signal de la plage d'action	non	non	non	non	non	non
Etat de chargement	non	non	non	non	non	non
Molette	non	non	non	non	oui	non
Indicateur d'aide	non	non	non	non	non	non
Vue Sm@rtClient	non	non	non	non	non	non
Visualisation/forçage	non	non	non	non	non	non
Vue de diagnostic système	non	non	non	non	non	non
Fenêtre de diagnostic système	non	non	non	non	non	non

Tableau 4-3 Dynamiquement

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Affichage	oui	oui	oui	oui	oui	oui
Commande opérateur	non	oui	oui	oui	oui	oui
Visibilité	oui	oui	oui	oui	oui	oui
Déplacements	oui	oui	oui	oui	oui	oui

Tableau 4-4 Objets supplémentaires

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Groupes	oui	oui	oui	oui	oui	oui
Touches programmables	oui	oui	oui	oui	oui	oui
Cycles	oui	oui	oui	oui	oui	oui
Scripts VB	non	oui	oui	oui	oui	oui
Listes de fonctions	oui	oui	oui	oui	oui	oui
Bibliothèque de graphiques	oui	oui	oui	oui	oui	oui

Tableau 4-5 Variables

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Variables multiplex	oui	oui	oui	oui	oui	oui
Tableaux	oui	oui	oui	oui	oui	oui
Types de données utilisateur	oui	oui	oui	oui	oui	oui
Interne	non	oui	oui	oui	oui	oui
Occurrences des types de données élémentaires	oui	oui	oui	oui	oui	oui
Occurrences de types de données utilisateur	oui	oui	oui	oui	oui	oui
Occurrence de tableaux	oui	oui	oui	oui	oui	oui

En outre, Openness prend en charge toutes les plages de valeurs prises en charge par les pilotes de communication.

Connexions

Openness prend en charge les connexions non intégrées prises en charge par les pupitres opérateur correspondants. Vous trouverez des informations complémentaires à ce sujet dans

l'aide en ligne de TIA Portal sous "Visualisation de processus > Communication avec les API > Dépendance par rapport à l'appareil".

Tableau 4-6 Listes

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Listes de textes	oui	oui	oui	oui	oui	oui
Listes de graphiques	oui	oui	oui	oui	oui	oui

Tableau 4-7 Textes

Objet	Basic Panels	Panels	Comfort Panels	Multi Panels	Mobile Panels	RT Advanced
Textes multilingues	oui	oui	oui	oui	oui	oui
Textes formatés et leurs occurrences	non	oui	oui	oui	oui	oui

4.5 Bibliothèques standard

Pour que les exemples de code fonctionnent, ajoutez les instructions Namespace suivantes au début de l'exemple de code correspondant :

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Online;
using Siemens.Engineering.Compiler; //for Compiler related methods
using Siemens.Engineering.Library;
using Siemens.Engineering.Library.Types;
using Siemens.Engineering.Library.MasterCopies;
using Siemens.Engineering.Compare;
using System.IO; //for certain export related methods
```

4.6 Remarques sur la performance de TIA Portal Openness V14

Objet racine

Vous pouvez indiquer plusieurs objets racine dans les fichiers d'importation.

Exemple : vous pouvez créer plusieurs listes de textes dans un fichier XML au lieu d'une liste de textes par fichier XML.

Fonctions Openness

Lorsque vous ouvrez une fonction Openness pour la première fois, l'appel peut durer plus longtemps que les appels suivants de cette fonction Openness.

Exemple : si vous exécutez plusieurs exportations de données de configuration les unes après les autres, la première exportation peut durer plus longtemps que les suivantes.

Introduction 5

Introduction

TIA Portal Openness V14 décrit des interfaces ouvertes pour l'ingénierie avec TIA Portal V14.

Avec TIA Portal Openness V14, vous pouvez automatiser l'ingénierie en commandant à distance le TIA Portal à partir d'un programme créé par vos soins.

TIA Portal Openness V14 vous permet d'exécuter les actions suivantes :

- Créer des données de projet
- Modifier des projets et des données de projet
- Supprimer des données de projet
- Lire des données de projet
- Mettre à disposition des projets et des données de projet pour d'autres applications.

Remarque

Siemens n'est pas responsable de la compatibilité des données transmises par le biais de ces interfaces avec un logiciel tiers et ne fournit aucune garantie à cet égard.

Nous attirons expressément l'attention sur le fait que l'utilisation inadéquate des interfaces peut entraîner la perte de données ou l'arrêt de la production.

Remarque

Les sections de code contenues dans cette documentation sont rédigées dans la syntaxe C#.

En raison de l'utilisation de sections de code courtes, une grande partie du traitement des erreurs n'est pas décrite.

Utilisation

Vous utilisez l'interface Openness aux fins suivantes :

- mettre à disposition des données de projet
- accéder au processus TIA Portal
- utiliser des données de projet

Utilisation de valeurs par défaut du domaine de l'automatisation

- par l'importation de données générées à distance
- par la commande à distance du portail TIA pour la génération de projets

Mise à disposition de données de projet du portail TIA pour des applications externes

- par l'exportation de données de projet

Conserver les atouts vis-à-vis de la concurrence par une ingénierie efficace

- Vous n'avez besoin de configurer les données d'ingénierie existantes dans le TIA Portal.
- Les processus d'ingénierie automatisés remplacent l'ingénierie manuelle.
- La réduction des coûts d'ingénierie renforcent la position concurrentielle.

Travail en commun sur des données de projet

- Les tests de routine et le traitement groupé des données peuvent avoir lieu parallèlement à la configuration en cours.

Voir aussi

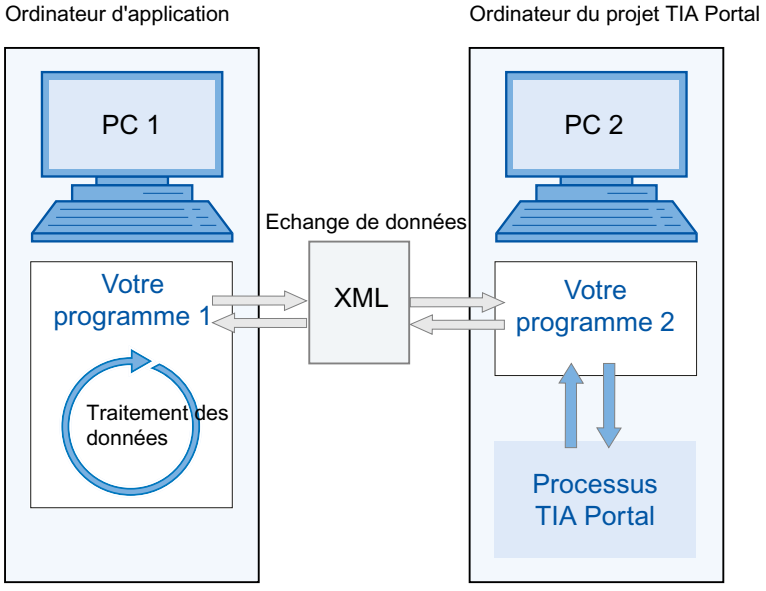
Configurations (Page 39)

Configurations

6

Vous pouvez travailler avec deux variantes de TIA Portal Openness V14 :

L'application et le TIA Portal se trouvent sur différents ordinateurs



- Les données sont échangées via des fichiers XML. Les fichiers XML peuvent être exportés ou importés par vos programmes.
- Les données exportées du portail TIA vers PC2 peuvent être modifiées sur PC1 et réimportées.

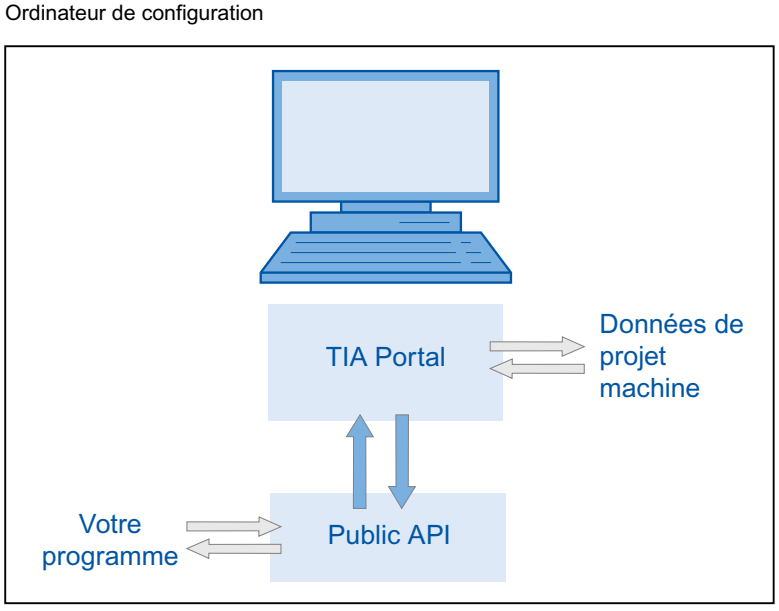
  Remarque

  Vous devez développer un programme exécutable "Votre programme 2" pour PC2, par exemple "programm2.exe". Le portail TIA s'exécute avec ce programme en arrière-plan.

  L'importation et l'exportation de fichiers XML s'effectuent exclusivement par le biais de l'API public.

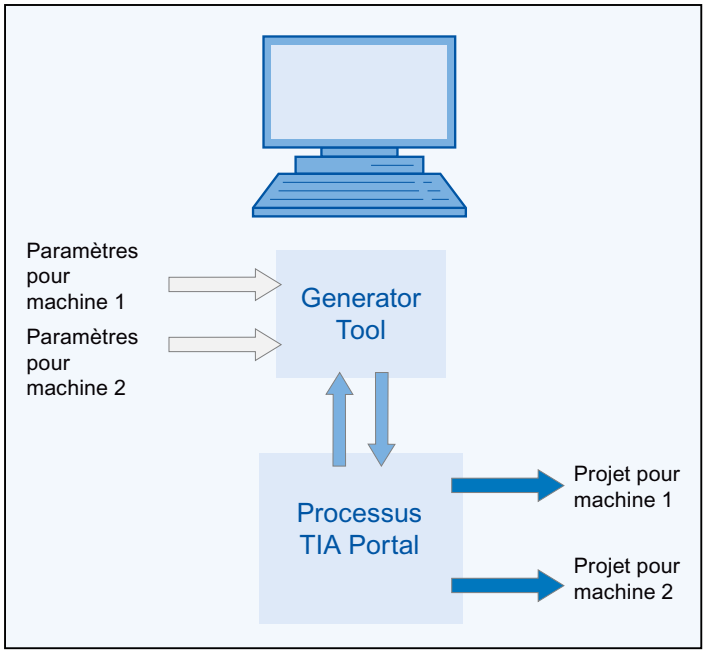
- Vous pouvez archiver les données échangées à des fins de vérification.
- Les données échangées peuvent être éditées à différents endroits et différents moments.

L'application et le TIA Portal se trouvent sur le même ordinateur



- Votre programme lance le TIA Portal avec ou sans interface utilisateur. Votre programme ouvre, enregistre et/ou ferme un projet. Le programme peut également établir une liaison avec un TIA Portal en cours d'exécution.
- Vous pouvez alors utiliser les fonctionnalités du TIA Portal pour demander, générer et modifier des données de projet ou pour déclencher des processus d'importation et d'exportation.
- Les données sont créées sous le contrôle du traitement du portail TIA et enregistrées dans les données du projet.

Un domaine d'application typique est la construction de machines modulaire.



- Un système d'automatisation efficace doit être appliqué sur des machines similaires.
- Un projet comprenant les composants de tous les modèles de machines est disponible dans le portail TIA.
- L'outil Generator Tool commande la création du projet correspondant à un modèle de machine donné.
- Il appelle les valeurs par défaut en lisant les paramètres du modèle de machine demandé.
- Il filtre les éléments concernés du projet global du portail TIA, les modifie le cas échéant et génère le projet de machine demandé.

Public API 7

7.1 Introduction

Vue d'ensemble

TIA Portal Openness prend en charge une sélection de fonctions pour des tâches définies, fonctions que vous pouvez appeler par le biais de Public API en-dehors de TIA Portal.

Remarque

Public API - Openness V13 SP1 et Openness V14

Si vous installez Openness V14, Openness V14 est installé à côté d'Openness V13 SP1.

Vous trouverez ci-après une vue d'ensemble des étapes de programmation typiques. Vous découvrirez comment les différentes sections de code interagissent et comment intégrer les différentes fonctions dans un programme complet. En outre, vous verrez quels éléments de code doivent être adaptés pour chaque tâche.

Exemple de programme

Les différentes étapes de programmation sont expliquées à l'aide de l'exemple de fonction "Créer l'accès de l'API dans une application console". Dans ce code de programme, vous intégrez les fonctions mises à disposition et vous adaptez les éléments de code correspondants pour cette tâche.

Fonctions

La rubrique ci-dessous indique les fonctions pour des tâches définies que vous pouvez appeler avec TIA Portal Openness en-dehors de TIA Portal.

Voir aussi

Possibilités d'utilisation (Page 27)

Liste d'objets (Page 29)

7.2 Etapes de programmation

Vue d'ensemble

TIA Portal Openness requiert les étapes de programmation suivantes pour l'accès via Public API :

1. Faire connaître TIA Portal dans l'environnement de développement
2. Configurer l'accès du programme à TIA Portal
3. Activer l'accès du programme à TIA Portal
4. Publier et démarrer TIA Portal
5. Ouvrir un projet
6. Exécuter des commandes
7. Enregistrer et fermer un projet
8. Mettre fin à la connexion au portail TIA

Remarque

Chaînes de caractères autorisées

Seuls certains caractères sont autorisés dans les chaînes de caractères sur TIA Portal. Toutes les chaînes de caractères transmises à TIA Portal par le biais de l'application Openness doivent se conformer à ces règles. Si vous transmettez un caractère non autorisé dans un paramètre, le système déclenche une exception.

Voir aussi

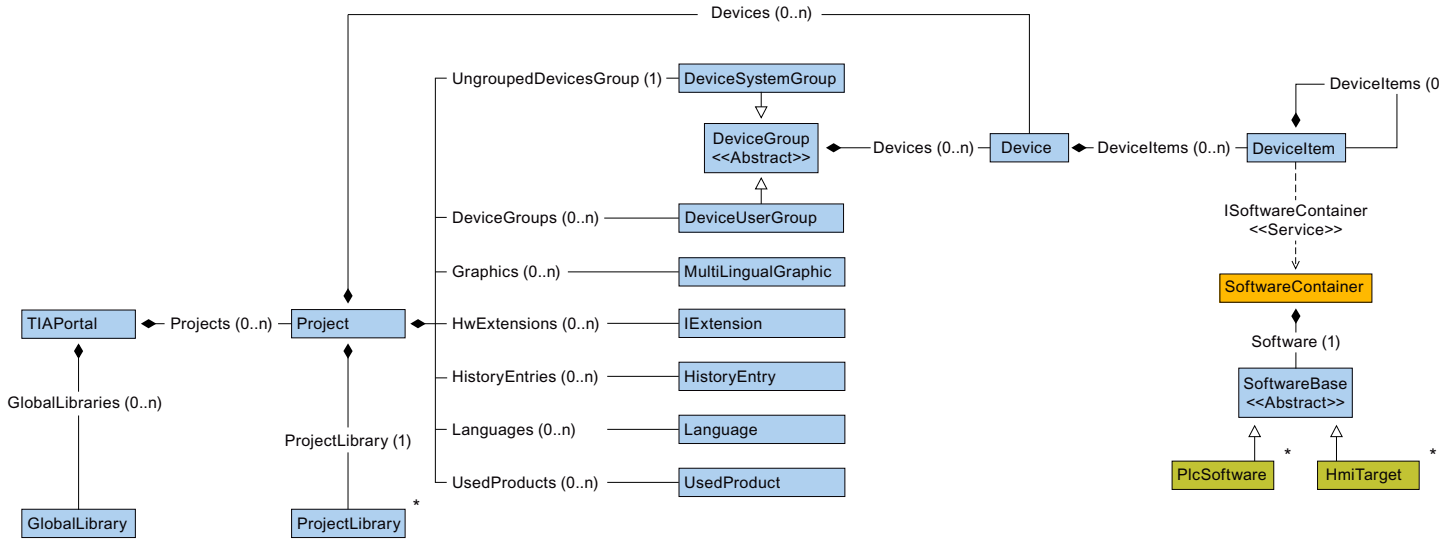
Exemple de programme (Page 62)

Utilisation des exemples de code (Page 67)

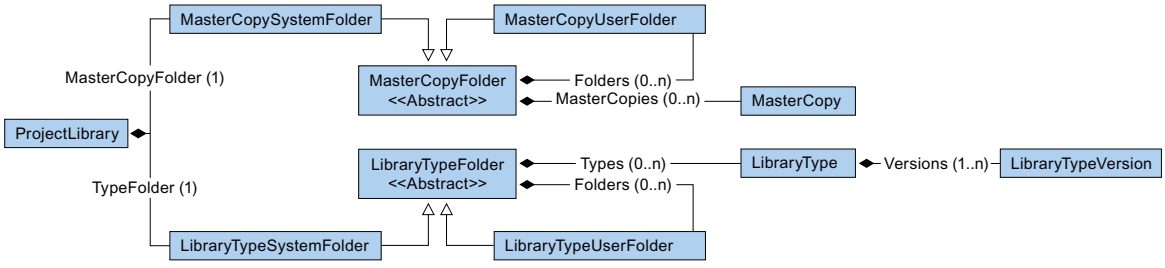
7.3 Modèle d'objet Openness V14

Vue d'ensemble

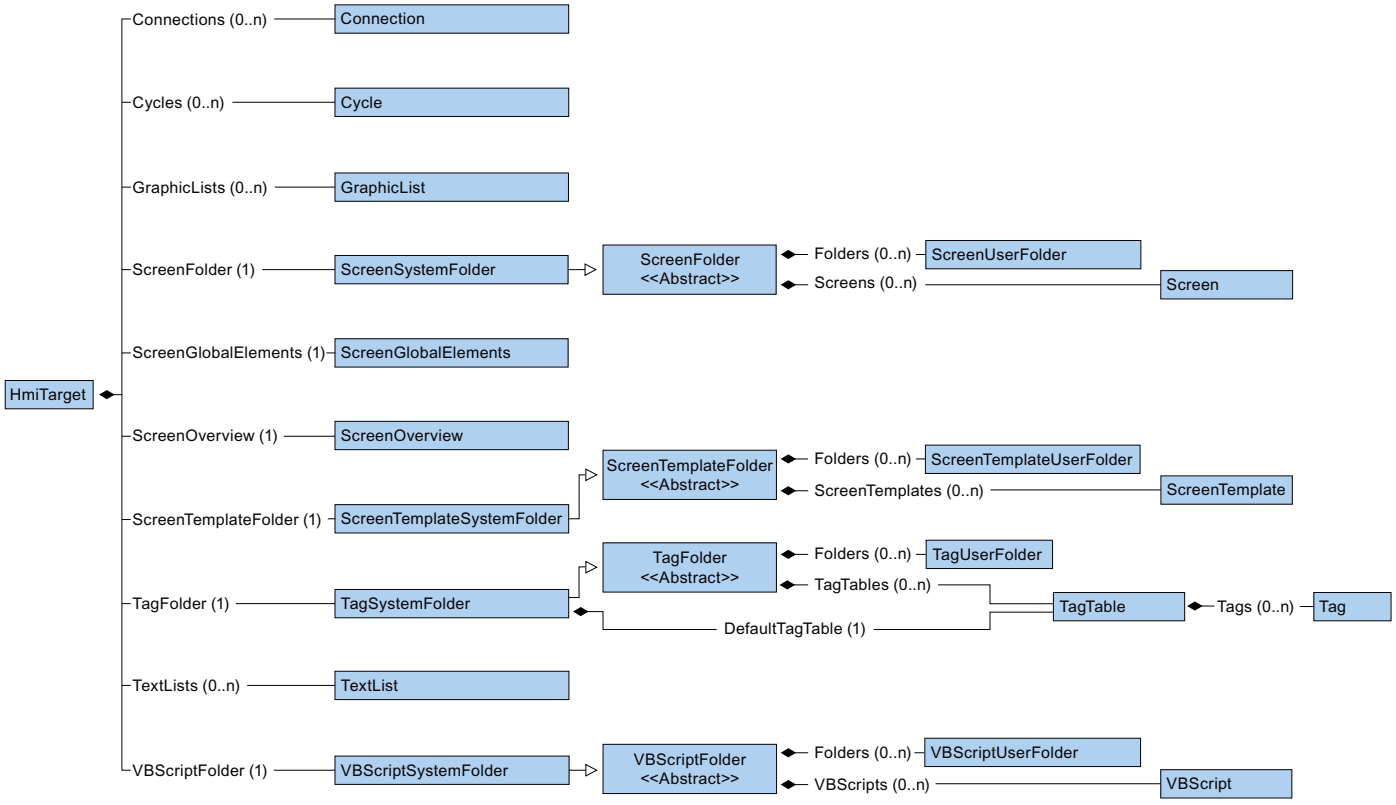
Le schéma suivant illustre le niveau le plus élevé du modèle d'objet Openness :



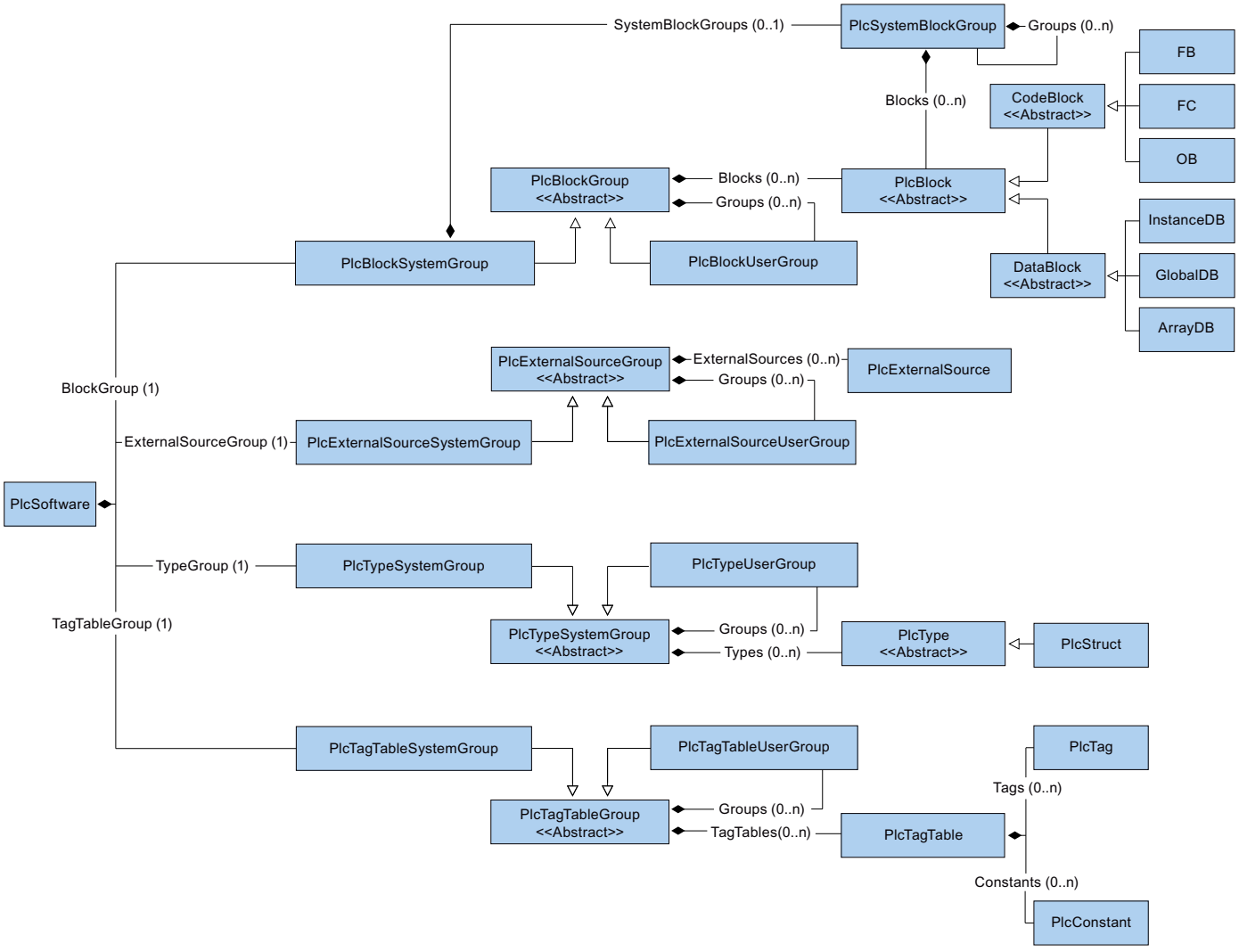
Le schéma suivant illustre les objets disponibles sous ProjectLibrary.



Le schéma suivant illustre les objets disponibles sous HmiTarget.



Le schéma suivant illustre les objets disponibles sous PlcSoftware.



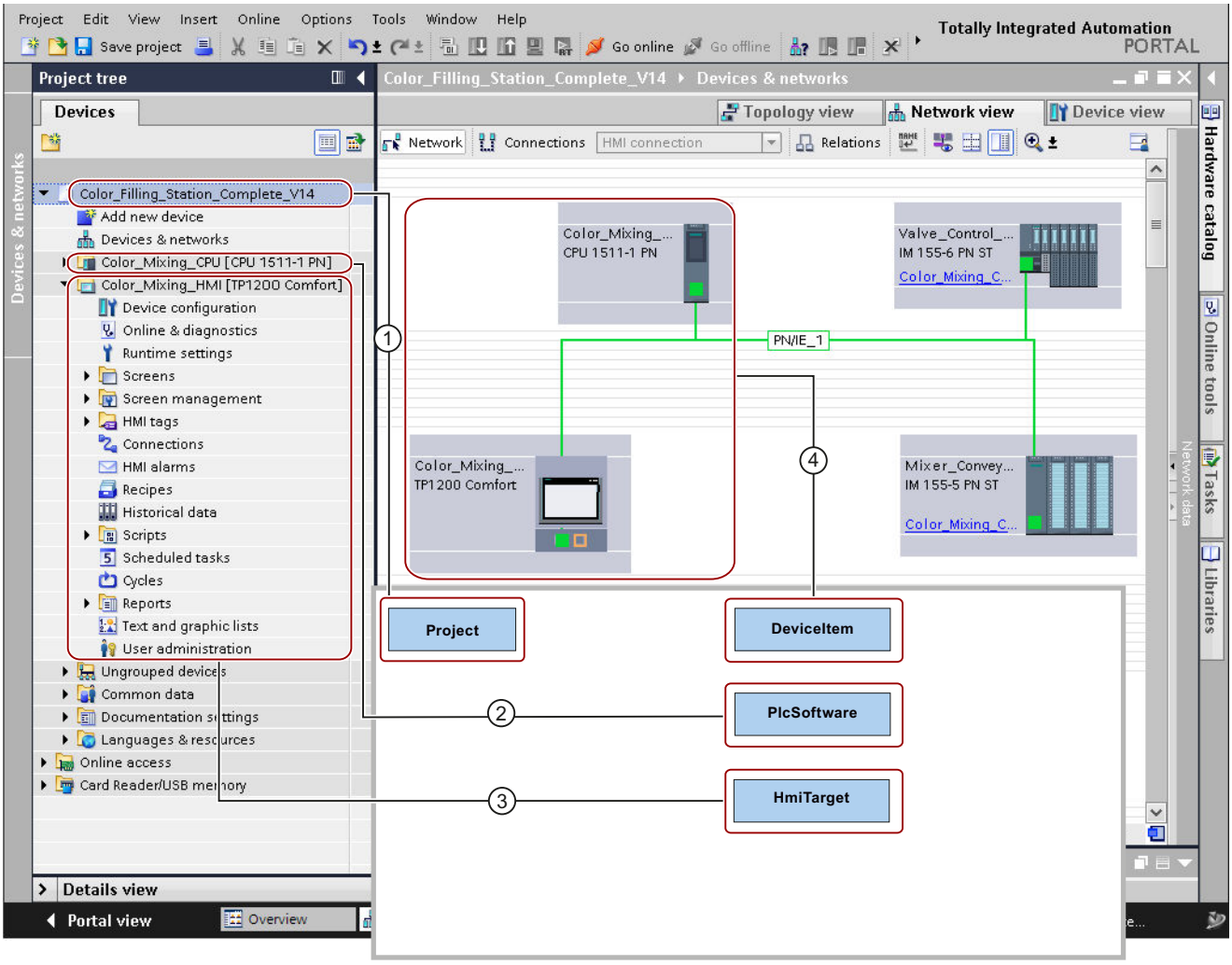
Accéder à des objets de listes

Pour adresser un objet dans une liste, vous avez les options suivantes :

- Adressez via l'index. Le comptage au sein des listes commence à partir de `0`.
- Utilisez la méthode `Find`.
- Utilisez cette méthode pour adresser un objet par son nom. Vous pouvez appliquer cette méthode à une composition ou une liste. La méthode `Find` n'est pas récursive.
  Exemple :
  ```
  ScreenComposition screens = folder.Screens;
  Screen screen = screens.Find("myScreen");
  ```
- Utilisez les noms symboliques.

Relation entre TIA Portal et le modèle d'objet Openness

La figure suivante présente la relation entre le modèle d'objet et un projet dans TIA Portal :



① L'objet "Project" correspond à un projet ouvert dans TIA Portal.

② L'objet "PlcSoftware" est de type "SoftwareBase"④ et correspond à un API. Le contenu de l'objet correspond à un API dans la navigation du projet, avec accès à des objets tels que des blocs ou des variables API.

③ L'objet "HmiTarget" est de type "SoftwareBase"④ et correspond à un pupitre opérateur. Le contenu de l'objet correspond à un appareil IHM dans la navigation du projet, avec accès à des objets tels que des vues ou des variables IHM.

④ L'objet "DeviceItem" correspond à un objet dans l'éditeur "Appareils & réseaux". Un objet du type "DeviceItem" peut être aussi bien un châssis qu'un module enfiché.

Voir aussi

Hiérarchie des objets matériels du modèle d'objet (Page 59)

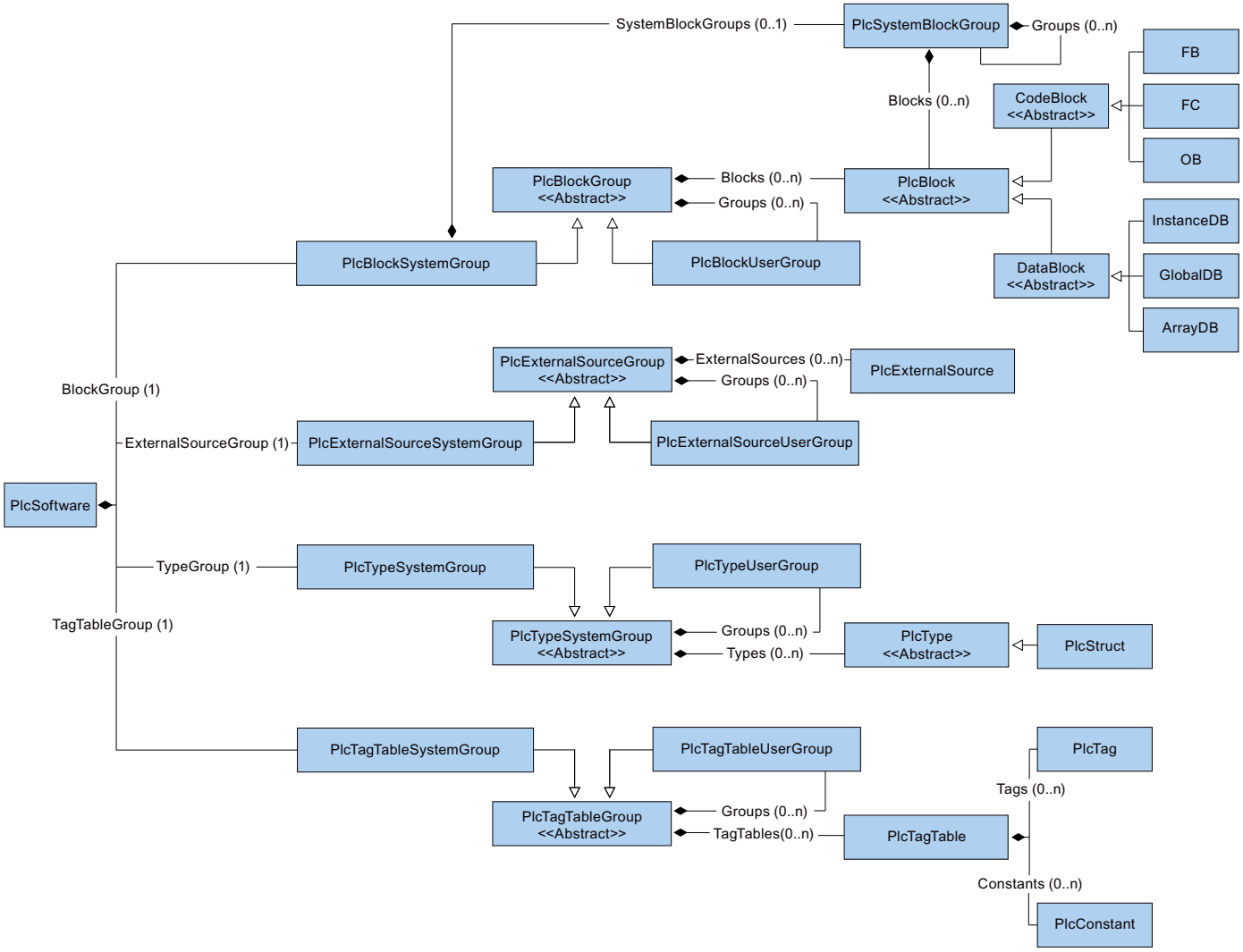
Public API (Page 43)

Exportation/importation (Page 199)
Ajouter un fichier externe (Page 182)
Exporter les graphiques d'un projet (Page 217)
Exporter des blocs (Page 257)
Exporter des blocs système (Page 262)
Exporter des tables de variables API (Page 264)
Importer une table de variables API (Page 265)
Exporter des variables ou constantes individuelles d'une table de variables API (Page 266)
Importer une seule variable ou constante dans une table de variables API (Page 267)
Exporter des connexions (Page 237)
Importation de connexions (Page 238)
Exportation de cycles (Page 219)
Importer des cycles (Page 220)
Exporter les listes de graphiques (Page 235)
Importer les listes de graphiques (Page 236)
Exporter des listes de textes à partir d'un appareil IHM (Page 232)
Importer une liste de texte dans un appareil IHM (Page 233)
Exporter des scripts VB (Page 229)
Importer des scripts VB (Page 231)
Exporter des tables de variables IHM (Page 221)
Importer une table de variables IHM (Page 224)
Exporter des variables individuelles d'une table de variables IHM (Page 225)
Importer des variables individuelles d'une table de variables IHM (Page 226)
Importer des graphiques dans un projet (Page 218)
Exporter toutes les vues d'un appareil IHM (Page 243)
Importer des vues dans un appareil IHM (Page 246)
Importer des modèles de vue (Page 254)
Exporter des modèles de vue à partir d'un dossier (Page 251)
Exporter une fenêtre permanente (Page 249)
Importer une fenêtre permanente (Page 250)

7.4 Blocs et types de modèle d'objet Openness

Introduction

Le schéma suivant représente le modèle de domaine des API afin de donner une vue d'ensemble de la structuration actuelle dans Openness.



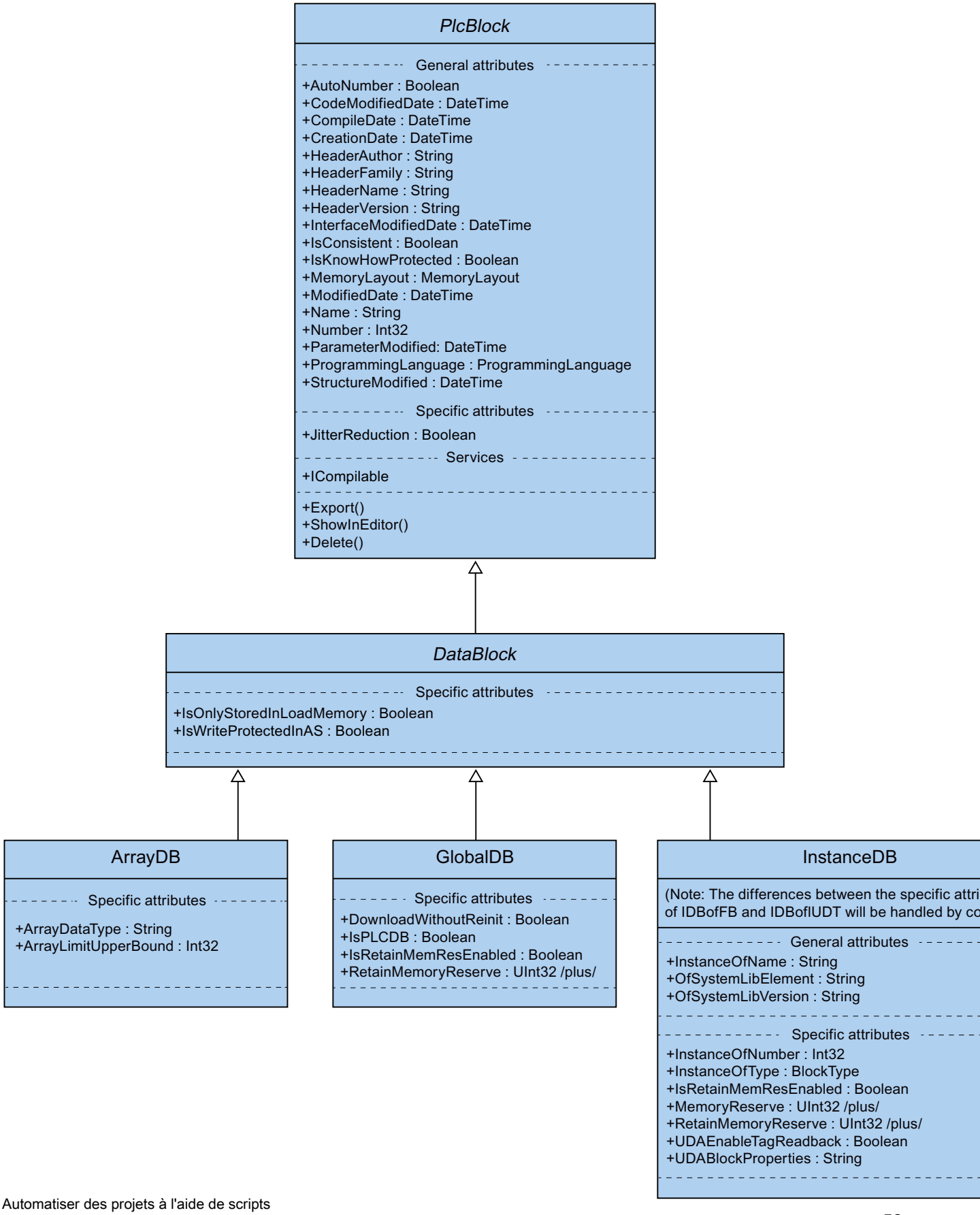
Représentation de blocs et types dans l'API Openness

L'élément de modèle simplifié des blocs et de la structure est basé sur les attributs dans l'API Openness. Les classes correspondantes mettent à disposition la fonction d'exportation ainsi que la fonction de compilation pour les blocs. ILibraryTypeInstance constitue l'unique interface pour ces éléments.

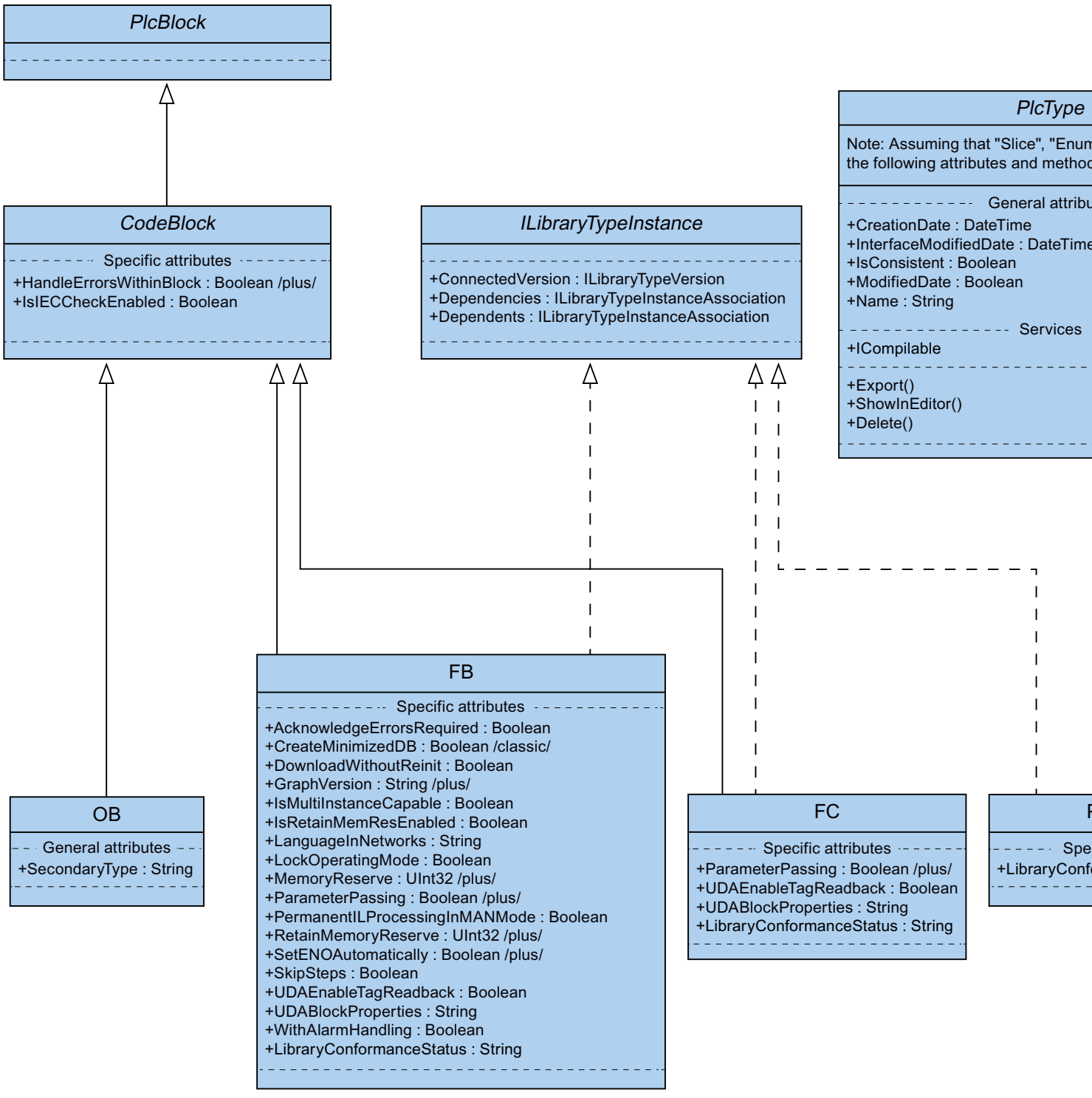
Diagrammes de classes

Toutes les classes qui ne sont pas directement instanciées sont définies de manière abstraite dans le modèle d'objet Openness.

Données


PlcBlock
General attributes
+AutoNumber : Boolean
+CodeModifiedDate : DateTime
+CompileDate : DateTime
+CreationDate : DateTime
+HeaderAuthor : String
+HeaderFamily : String
+HeaderName : String
+HeaderVersion : String
+InterfaceModifiedDate : DateTime
+IsConsistent : Boolean
+IsKnowHowProtected : Boolean
+MemoryLayout : MemoryLayout
+ModifiedDate : DateTime
+Name : String
+Number : Int32
+ParameterModified: DateTime
+ProgrammingLanguage : ProgrammingLanguage
+StructureModified : DateTime
Specific attributes
+JitterReduction : Boolean
Services
+ICompilable
+Export()
+ShowInEditor()
+Delete()
DataBlock
Specific attributes
+IsOnlyStoredInLoadMemory : Boolean
+IsWriteProtectedInAS : Boolean
ArrayDB
Specific attributes
+ArrayDataType : String
+ArrayLimitUpperBound : Int32
GlobalDB
Specific attributes
+DownloadWithoutReinit : Boolean
+IsPLCDB : Boolean
+IsRetainMemResEnabled : Boolean
+RetainMemoryReserve : UInt32 /plus/
InstanceDB
(Note: The differences between the specific attri
of IDBofFB and IDBofIUDT will be handled by co
General attributes
+InstanceOfName : String
+OfSystemLibElement : String
+OfSystemLibVersion : String
Specific attributes
+InstanceOfNumber : Int32
+InstanceOfType : BlockType
+IsRetainMemResEnabled : Boolean
+MemoryReserve : UInt32 /plus/
+RetainMemoryReserve : UInt32 /plus/
+UDAEnableTagReadback : Boolean
+UDABlockProperties : String

Code et type



Espaces de noms

Comme résultat de la nouvelle structure du modèle d'objet, les espaces de noms ont été modifiés.

L'instruction d'espace de noms Siemens.Engineering.SW existante a été étendue et modifiée comme suit :

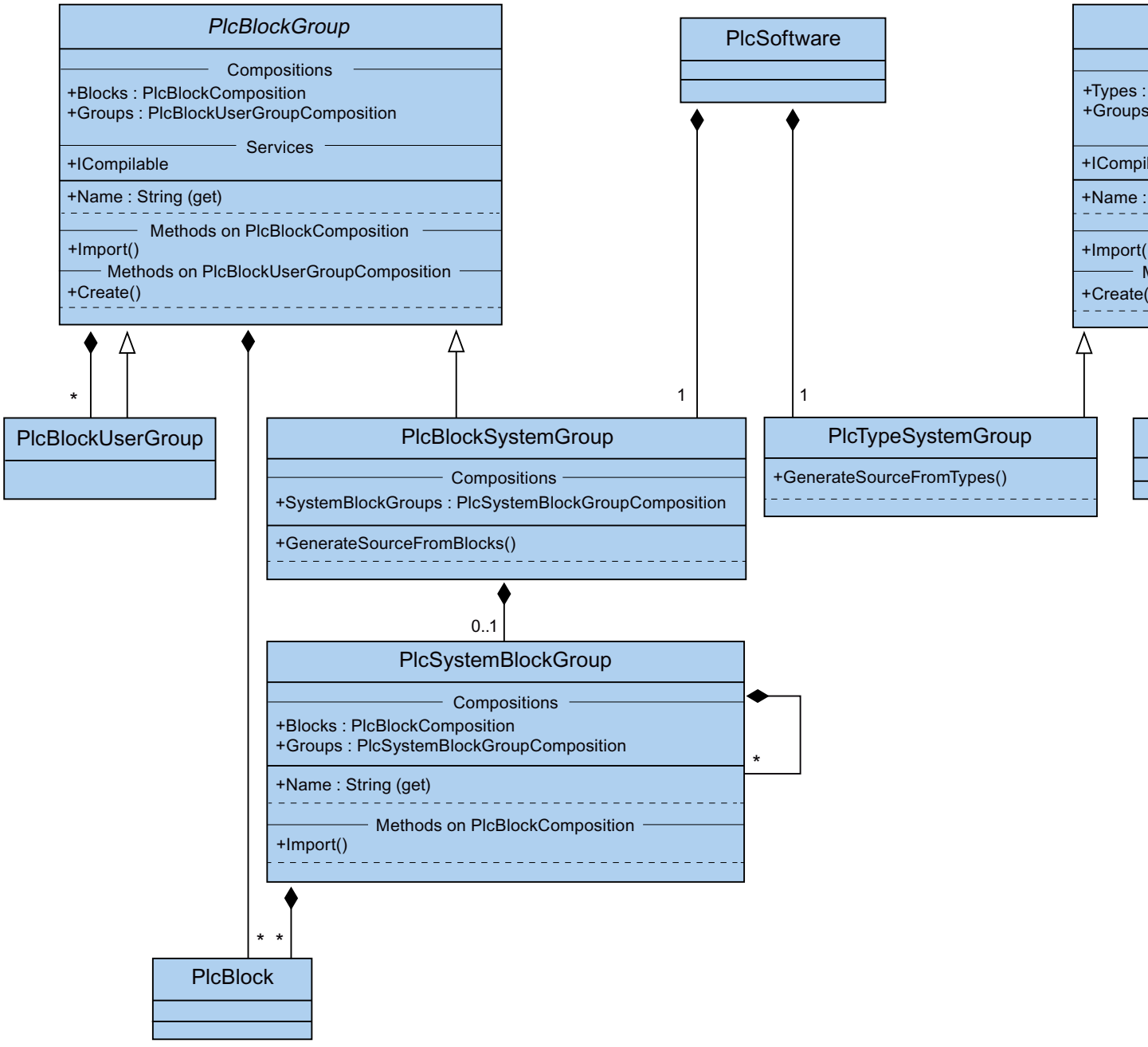
- ControllerTarget est modifié dans PlcSoftware.
- "Blocks" est ajouté avec le contenu suivant :
  - ArrayDB
  - CodeBlock
  - CompileUnit
  - CompileUnitComposition
  - DataBlock
  - FB
  - FC
  - GlobalDB
  - InstanceDB
  - PlcBlock
  - PlcBlockComposition
  - OB
  - MemoryLayout
  - ProgrammingLanguage
  - PlcBlockGroup
  - PlcBlockSystemGroup
  - PlcBlockUserGroup
  - PlcBlockUserGroupComposition
  - PlcSystemBlockGroup
  - PlcSystemBlockGroupComposition

- "ExternalSources" est ajouté avec le contenu suivant :
  - PlcExternalSource
  - PlcExternalSourceComposition
  - PlcExternalSourceGroup
  - PlcExternalSourceSystemGroup
  - PlcExternalSourceUserGroup
  - PlcExternalSourceUserGroupComposition
- "Types" est ajouté avec le contenu suivant :
  - PlcType
  - PlcTypeComposition
  - PlcStruct
  - PlcTypeGroup
  - PlcTypeSystemGroup
  - PlcTypeUserGroup
  - PlcTypeUserGroupComposition

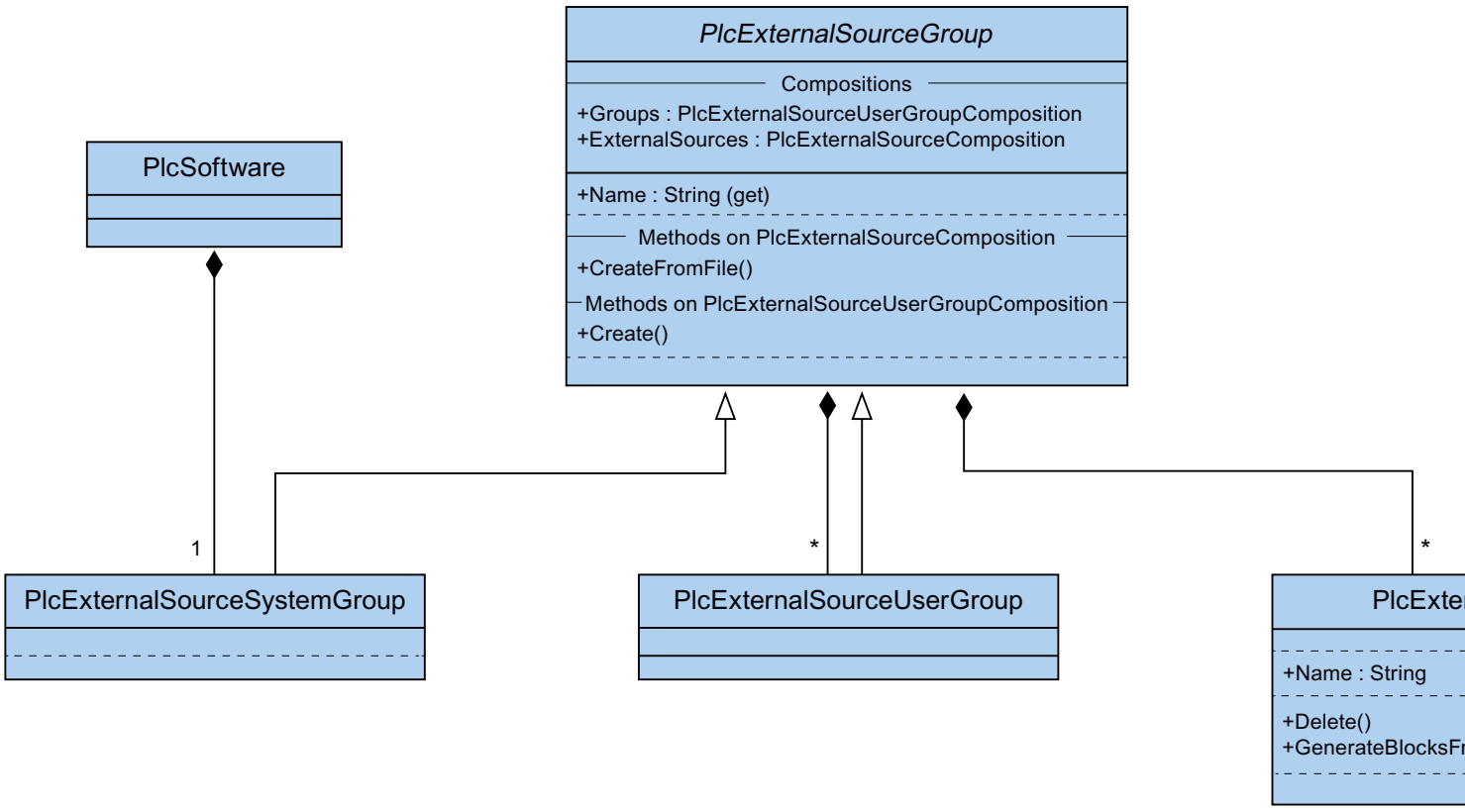
Représentation de groupes de blocs et types dans l'API Openness

Les deux groupes "PlcBlocks" du niveau supérieur ("Blocs de programme" dans l'interface utilisateur de TIA Portal) et "PlcTypes" ("Types de données API" dans l'interface utilisateur de TIA Portal) contiennent des blocs et des définitions de type. Ces groupes mettent la fonction d'importation et la fonction de compilation à la disposition des blocs. Les noms de classes avec le préfixe "a.k.a." correspondent aux noms utilisés dans le modèle d'objet Openness V13 SP1. Étant donné que la plupart des méthodes qui s'appliquent aux fonctionnalités des groupes ne sont accessibles que via des bibliothèques, il existe une représentation "intégrée" ou "condensée" des bibliothèques et des méthodes correspondantes dans les classes "Host".

Blocs et types



Sources externes



7.5 Hiérarchie des objets matériels du modèle d'objet

Relation entre les éléments visibles du portail TIA et les éléments structurés du modèle d'objet

Modèle matériel	Explication
Appareil (Device)	Objet conteneur pour une configuration centralisée ou décentralisée.
Élément d'appareil (IDeviceItem)	Chaque objet élément d'appareil possède un objet conteneur. La relation logique est "Éléments".

Pour les objets éléments d'appareil, la relation conteneur-élément est similaire à la relation des modules.

Exemple : un appareil contient un ou plusieurs emplacements. Un emplacement contient des modules. Un module contient des sous-modules.

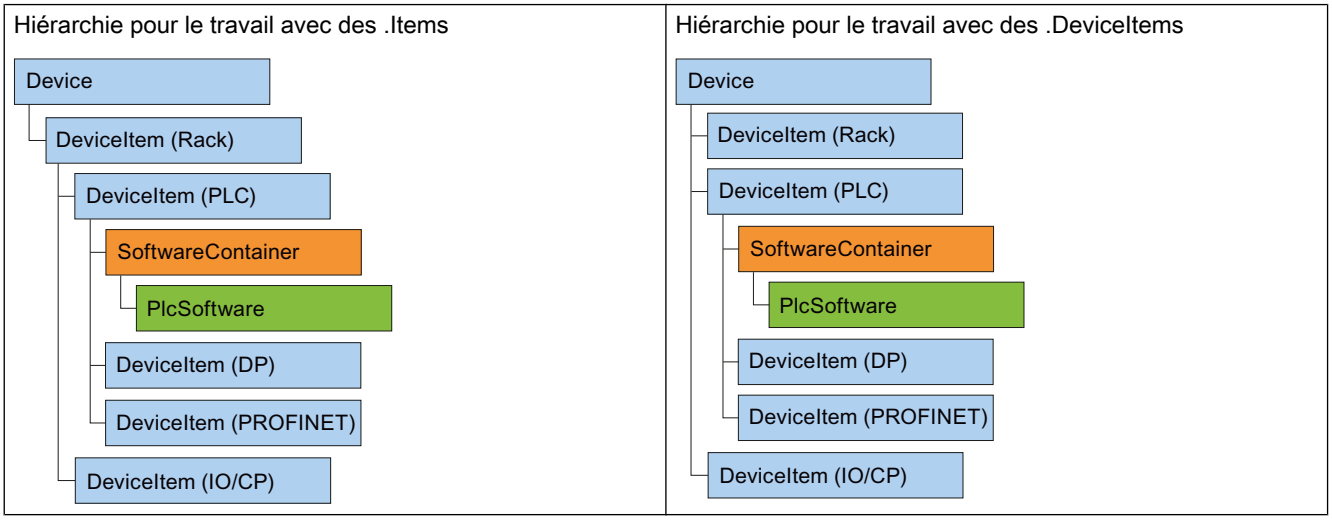
Il s'agit de la relation similaire à la représentation dans la vue de réseau et la vue d'appareil de TIA Portal. L'attribut "PositionNumber" d'un élément d'élément d'appareil est unique dans la zone du conteneur.

La relation enfant-parent entre les objets élément d'appareil est une relation logique pure dans le modèle d'objet. Un enfant ne peut exister sans ses parents.

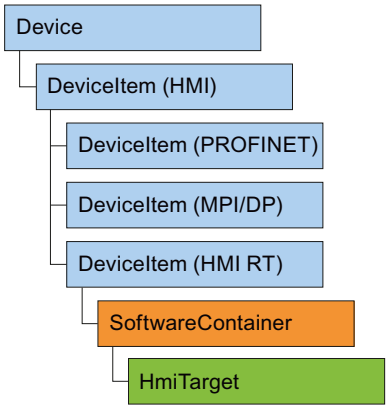
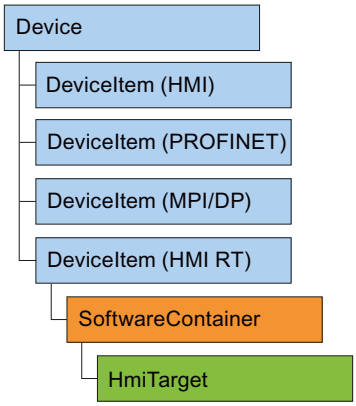
- Si un sous-module est structuré comme une partie d'un module (enfant), le sous-module ne peut pas être retiré sans le module.
- Si vous ajoutez au module un sous-module, puis vous le retirez du sous-module, cet enfant a les mêmes parents que le module.

Le diagramme ci-dessous indique la hiérarchie entre des appareils et des éléments d'appareils API et IHM.

Hiérarchie entre appareils API



Hiérarchie entre appareils IHM

Hiérarchie pour le travail avec des .Items	Hiérarchie pour le travail avec des .DeviceItems
	

Voir aussi

Public API (Page 43)

Exportation/importation (Page 199)

7.6 Informations sur les versions d'Openness installées

Conditions

- Openness et TIA Portal sont installés

Utilisation

À partir d'Openness V14, chaque version installée dispose d'une clé de Registre qui contient des informations sur la version. Cela permet de créer automatiquement le fichier de configuration d'application pour chaque version d'Openness installée.

La clé de Registre se trouve dans le chemin suivant :

HKEY_LOCAL_MACHINE\Software\Siemens\Automation\Openness\14.0\PublicAPI

Remarque

Le numéro de version présent dans ce chemin est toujours identique au numéro de la version de TIA Portal actuelle installée. En cas de plusieurs installations parallèles, plusieurs jeux d'entrées correspondant à Openness sont présents dans le registre.

Il existe une clé unique par version d'Openness. Les noms des versions sont identiques à ceux dans la description de Assembly, par ex. les entrées de registre pour Openness V14 :

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\PublicAPI
\14.0.0.0]"PublicKeyToken"="d29ec89bac048f84"
"Siemens.Engineering"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.dll"
"Siemens.Engineering.Hmi"="C:\Program Files\Siemens\Automation\Portal V14\PublicAPI
\V14\Siemens.Engineering.Hmi.dll"
"EngineeringVersion"="V14"
"AssemblyVersion"="14.0.0.0"
```

Remarque

Si vous souhaitez créer un fichier de configuration d'application (Page 69), vous pouvez connaître le chemin d'accès à Siemens.Engineering.dll, Siemens.Engineering.Hmi.dll et au jeton de la clé publique grâce à la clé de Registre.

7.7 Exemple de programme

Exemple d'application : Créer l'accès de l'API dans une application

Le code du programme complet de l'exemple d'application est indiqué ci-après. Les étapes typiques de programmation sont décrites ci-après à l'aide de cet exemple.

Remarque

Un fichier de configuration d'exemple (Page 69) est requis pour l'exemple d'application.

```
using System;
using Siemens.Engineering;
using Siemens.Engineering.HW;
using Siemens.Engineering.SW;
using Siemens.Engineering.SW.Blocks;
using Siemens.Engineering.SW.ExternalSources;
using Siemens.Engineering.SW.Tags;
using Siemens.Engineering.SW.Types;
using Siemens.Engineering.Hmi;
using HmiTarget = Siemens.Engineering.Hmi.HmiTarget;
using Siemens.Engineering.Hmi.Tag;
using Siemens.Engineering.Hmi.Screen;
using Siemens.Engineering.Hmi.Cycle;
using Siemens.Engineering.Hmi.Communication;
using Siemens.Engineering.Hmi.Globalization;
using Siemens.Engineering.Hmi.TextGraphicList;
using Siemens.Engineering.Hmi.RuntimeScripting;
using System.Collections.Generic;
using Siemens.Engineering.Compiler; //for Compiler related methods
using Siemens.Engineering.Library; //for library and update check related methods
using System.IO; //for certain export related methods


namespace HelloTIA
{
    internal class Program
    {
        private static void Main(string[] args)
        {
            RunTiaPortal();
        }

        private static void RunTiaPortal()
        {
            Console.WriteLine("Starting TIA Portal");
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                Console.WriteLine("TIA Portal has started");
                ProjectComposition projects = tiaPortal.Projects;

                Console.WriteLine("Opening Project...");

                string projectPath = @"C:\Demo\AnyCompanyProject.ap14";//edit the path
according to your project
                Project project = null;
                try
                {
                    project = projects.Open(projectPath);
                }
                catch (Exception)
                {
                    Console.WriteLine(String.Format("Could not open project {0}",
projectPath));
                    Console.WriteLine("Demo complete hit enter to exit");
                    Console.ReadLine();
```

```
                    return;
                }

                Console.WriteLine(String.Format("Project {0} is open", project.Path));

                IterateThroughDevices(project);

                project.Close();

                Console.WriteLine("Demo complete hit enter to exit");
                Console.ReadLine();
            }
        }

        private static void IterateThroughDevices(Project project)
        {
            if (project == null)
            {
                Console.WriteLine("Project cannot be null");
                return;
            }

            Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));

            foreach (Device device in project.Devices)
            {
                Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
            }

            Console.WriteLine();
        }
    }
}
```

Procédure par étapes

1. Faire connaître le portail TIA dans l'environnement de développement

Créez dans votre environnement de développement un renvoi à tous les "fichiers dll" dans le répertoire "C:\Program Files\Siemens\Automation\PortalV14\PublicAPI\V14".

Cette procédure est illustrée ci-après pour le fichier "Siemens.Engineering.dll".

Le fichier "Siemens.Engineering.dll" se trouve dans le répertoire "C:\Program Files\Siemens \Automation\PortalV14\PublicAPI\V14". Créez dans votre environnement de développement un renvoi au fichier "Siemens.Engineering.dll".

Remarque

Affectez la valeur "False" au paramètre "CopyLocal" dans les propriétés de référence.

2. Publier la plage de noms pour le TIA Portal

Insérez le code suivant :

```
using Siemens.Engineering;
```

3. Publier et démarrer TIA Portal

Pour publier et démarrer TIA Portal, insérez le code suivant :

```
using (TiaPortal tiaPortal = new TiaPortal())
```

4. Ouvrir un projet

Pour ouvrir un projet, vous pouvez par exemple utiliser le code suivant :

```
ProjectComposition projects = tiaPortal.Projects;
Console.WriteLine("Opening Project...");
string projectPath = @"C:\Demo\AnyCompanyProject.ap14";
Project project = null;
try
{
    project = projects.Open(projectPath);
}
catch (Exception)
{
    Console.WriteLine(String.Format("Could not open project {0}", projectPath));
    Console.WriteLine("Demo complete hit enter to exit");
    Console.ReadLine();
    return;
}
Console.WriteLine(String.Format("Project {0} is open", project.Path));
```

5. Énumérer les appareils d'un projet

Ajoutez le code suivant pour énumérer tous les appareils du projet :

```
static private void IterateThroughDevices(Project project)
    {
        if (project == null)
        {
            Console.WriteLine("Project cannot be null");
            return;
        }

        Console.WriteLine();
        Console.WriteLine(String.Format("Iterate through {0} device(s)",
project.Devices.Count));
        foreach (Device device in project.Devices)
        {
            Console.WriteLine(String.Format("Device: \"{0}\".", device.Name));
        }
        Console.WriteLine();
    }
```

6. Enregistrer et fermer un projet

Insérez le code suivant puis enregistrez et fermez le projet :

```
project.Save();
project.Close();
```

7.8 Utilisation des exemples de code

Structure des sections de code

Chaque section de code de cette documentation est réalisée comme fonction sans valeur de retour avec une référence d'objet comme paramètre de transmission. L'accès aux objets de TIA Portal se fait par leur nom à l'aide de la méthode `Find`.

Dans la section de code qui suit, une vue "MyScreen" du groupe "myScreenFolder" est supprimée :

```
//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    //Change the names according to the names in your project
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

Pour exécuter cette section de code, les éléments suivants sont requis :

- Un projet WinCC avec un appareil IHM contenant un groupe avec au moins une vue.
- Une fonction qui est instanciée par le pupitre opérateur.

Remarque

Si vous indiquez des chemins de répertoire, utilisez le chemin absolu, par ex. "C:/path/file.txt".

Les chemins de répertoire relatifs ne sont autorisés que pour les fichiers XML et l'importation/ l'exportation, par ex. "file.txt" ou "C:/path01/.../path02/file.txt".

Exemple d'exécution de la section de code

Pour exécuter la section de code "DeleteScreenFromFolder" dans le cadre de l'exemple de programme "Hello TIA", utilisez l'exemple suivant :

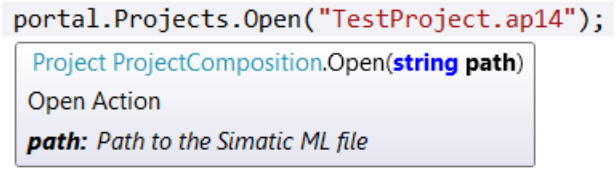
```
//In the sample program "Hello TIA" replace the function call
//"IterateThroughDevices(project)" by the following functions calls:
    HmiTarget hmiTarget = GetTheFirstHmiTarget(project);
    DeleteScreenFromFolder(hmiTarget);

//Put the following function definitions before or after the
//function definition of "private static void IterateThroughDevices(Project project)":
private static HmiTarget GetTheFirstHmiTarget(Project project)
{
    if (project == null)
    {
        Console.WriteLine("Project cannot be null");
        throw new ArgumentNullException("project");
    }
    foreach (IDevice device in project.Devices)
    {
        foreach (IDeviceItem deviceItem in device.DeviceItems)
        {
            DeviceItem deviceItemToGetService = deviceItem as DeviceItem;
            SoftwareContainer container =
deviceItemToGetService.GetService<ISoftwareContainer>() as SoftwareContainer;
            if (container != null)
            {
                HmiTarget hmi = container.Software as HmiTarget;
                if (hmi != null)
                {
                    return hmi;
                }
            }
        }
    }
    return null;
}

//Deletes a single screen from a user folder or a system folder
private static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    string screenName = "MyScreen";
    ScreenUserFolder folder = hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Delete();
    }
}
```

7.9 Fonctions générales

7.9.1 IntelliSense-Support pour Openness

Utilisation

IntelliSense-Support pour Openness vous offre de l'aide sur les propriétés ou méthodes existantes par le biais d'info-bulles et peut vous renseigner sur le nombre, les noms et les types de paramètres requis. Dans l'exemple suivant, le paramètre en gras dans la première ligne indique le paramètre requis suivant lors de l'entrée de la fonction.



Vous pouvez appeler manuellement des informations sur les paramètres de différentes manières : Cliquez sur "Edit IntelliSense/Parameter Info", à l'aide de la combinaison de touches <CTRL + MAJ + ESPACE> ou cliquez sur le bouton "Parameter Info" dans la barre d'outils de l'éditeur.

7.9.2 Etablissement d'une connexion au portail TIA

Introduction

Vous démarrez TIA Portal avec TIA Portal Openness ou établissez la liaison avec un TIA Portal en cours d'exécution. Si vous démarrez TIA Portal avec TIA Portal Openness, indiquez si TIA Portal doit être démarré avec ou sans interface utilisateur graphique. Si vous travaillez avec TIA Portal sans interface utilisateur, TIA Portal est uniquement lancé comme un processus du système d'exploitation. Si nécessaire, une application Openness vous permet de créer plusieurs instances de TIA Portal.

Remarque

Si vous utilisez l'application Openness pour accéder à l'interface TIA Portal, l'éditeur ne doit pas être un éditeur IHM. Vous pouvez ouvrir l'éditeur "Appareils & réseaux" ou l'éditeur de programmation manuellement ou via Public API.

Vous disposez des options suivantes pour démarrer TIA Portal avec une application Openness.

- Vous pouvez utiliser un fichier de configuration d'application (recommandé pour la plupart des applications).
- Utilisez la méthode "AssemblyResolve" (recommandé pour la copie, etc.).

Remarque

Nous vous recommandons de charger Siemens.Engineering.dll à l'aide du fichier de configuration d'application. Cette méthode permet de tenir compte des noms forts et les modifications dommageables au Engineering.dll entraînent une erreur de chargement. Ce qui n'est pas détectable en cas d'utilisation de la méthode AssemblyResolve.

Démarrage de TIA Portal avec un fichier de configuration d'application

Créez dans le fichier de configuration d'application des renvois à toutes les bibliothèques de programmes requises. Répartissez le fichier de configuration d'application avec l'application Openness.

Enregistrez le fichier de configuration d'application "app.config" dans le même répertoire que l'application Openness et intégrez ce fichier dans votre application. Vérifiez dans chaque code si le chemin d'accès au fichier correspond au chemin d'installation de TIA Portal.

Vous pouvez utiliser l'extrait de code suivant pour le fichier de configuration d'application :

```
<?xml version="1.0"?>
<configuration>
    <runtime>
        <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installation -->
              <codeBase version="14.0.0.0" href="FILE://C:\Program Files\Siemens\Automation
\Portal V14\PublicAPI\V14\Siemens.Engineering.dll"/>
            </dependentAssembly>
            <dependentAssembly>
                <assemblyIdentity name="Siemens.Engineering.Hmi" culture="neutral"
publicKeyToken="d29ec89bac048f84"/>
                <!-- Edit the following path according to your installation -->
              <codeBase version="14.0.0.0" href="FILE://C:\Program Files\Siemens\Automation
\Portal V14\PublicAPI\V14\Siemens.Engineering.Hmi.dll"/>
            </dependentAssembly>
        </assemblyBinding>
    </runtime>
</configuration>
```

Pour ouvrir une nouvelle instance de TIA Portal via le fichier de configuration d'application, utilisez le code de programme suivant :

```
//Connect an openness application via API using
using System;
using System.IO;
using Siemens.Engineering;

namespace UserProgram
{
    internal class MyProgram
    {
        public static void Main(string[] args)
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface)
            //
            // To start TIA Portal without user interface:
           // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithoutUserInterface)
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
    }
}
```

Démarrage de TIA Portal avec la méthode "AssemblyResolve"

Créez le code de programme d'Openness de telle sorte que l'enregistrement sur l'événement "AssemblyResolve" soit effectué le plus tôt possible. Encapsulez l'accès à TIA Portal dans un objet ou une méthode supplémentaire.

La prudence est de mise lors de la résolution d'Engineering Assembly par la méthode AssemblyResolve. Lorsque des types de l'Engineering Assembly sont utilisés avant l'exécution de l'Assembly Resolver, le programme se bloque. Cela est dû au fait que le compilateur Just-in-time (compilateur JIT) ne compile une méthode qu'au moment où il doit l'exécuter. Lorsque des types d'un Engineering Assembly sont utilisés par ex. dans "Main", le compilateur JIT essaie de compiler "Main" pendant l'exécution du programme. Ce qui échoue parce que le compilateur JIT ne sait pas où trouver l'Engineering Assembly. L'enregistrement de Assembly Resolver dans Main n'y change rien. La méthode doit être en cours d'exécution avant l'enregistrement de Assembly Resolver et compilée avant que celui-ci ne puisse être exécuté. La solution à ce problème consiste à placer la Business Logic, qui utilise les types provenant de l'Engineering Assembly, dans une méthode séparée. Et la méthode séparée utilise uniquement des types que le compilateur JIT comprend déjà. L'exemple suivant utilise une méthode qui fournit en retour "void", ne possède aucun paramètre et contient toutes les Business Logic. Le compilateur JIT peut maintenant compiler "Main" avec succès parce qu'il comprend tous les types présents dans Main. Assembly Resolver est déjà enregistré lorsque RunTiaPortal est appelé pendant l'exécution. Ainsi, le compilateur JIT sait où trouver l'Engineering Assembly lorsqu'il cherche les types de Business Logic.

Pour ouvrir une nouvelle instance de TIA Portal, utilisez le code de programme suivant.

```
using System;
using System.IO;
using System.Reflection;
using Siemens.Engineering;

namespace UserProgram
{
    static class MyProgram
    {
        public static void Main(string[] args)
        {
            AppDomain.CurrentDomain.AssemblyResolve += MyResolver;
            RunTiaPortal();
        }
        private static void RunTiaPortal()
        {
            // To start TIA Portal with user interface:
            // using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface)
            //
            // To start TIA Portal without user interface:
            // using (TiaPortal tiaPortal = new
TiaPortal(TiaPortalMode.WithoutUserInterface)
            using (TiaPortal tiaPortal = new TiaPortal(TiaPortalMode.WithUserInterface))
            {
                //begin of code for further implementation
                //...
                //end of code
            }
        }
        private static Assembly MyResolver(object sender, ResolveEventArgs args)
        {
            int index = args.Name.IndexOf(',');
            if (index == -1)
            {
                return null;
            }
            string name = args.Name.Substring(0, index) + ".dll";
            string path = Path.Combine(@"C:\Program Files\Siemens\Automation\Portal
V14\PublicAPI\V14\", name);
            // User must provide the correct path
            string fullPath = Path.GetFullPath(path);
            if (File.Exists(fullPath))
            {
                return Assembly.LoadFrom(fullPath);
            }
            return null;
        }
    }
}
```

Accéder à des instances actives de TIA Portal

Pour pouvoir établir une liaison à une instance active de TIA Portal avec une application Openness, énumérez d'abord les instances de TIA Portal. Vous pouvez vous connecter à plusieurs instances au cours d'une session Windows. L'instance active peut être TIA Portal avec ou sans interface utilisateur démarrée :

```
foreach (TiaPortalProcess tiaPortalProcess in TiaPortal.GetProcesses()
    {
        ...
    }
```

Si vous connaissez l'ID de processus de l'instance du TIA Portal, utilisez cet ID de processus pour accéder à l'objet. Le démarrage de TIA Portal nécessite un certain temps avant que vous ne puissiez le connecter à Openness.

Lors de l'établissement de la liaison à une instance active de TIA Portal, une invite de commande du pare-feu Openness s'affiche. Vous pouvez choisir parmi ces options pour la liaison :

- Autoriser la liaison de manière unique
- Ne pas autoriser la liaison
- Toujours autoriser les liaisons de cette application
  Pour plus d'informations, voir Pare-feu Openness (Page 74).

Remarque

Si l'invite de commande du registre est rejetée trois fois, le système déclenche une exception du type `EngineeringSecurityException`.

Une fois la liaison au processus établie, vous pouvez appeler des informations sur les instances de TIA Portal à l'aide de l'un des attributs suivants :

Attribut	Information
`InstalledSoftware as IList <TIAPortalProduct>`	Fournit en retour des informations sur les produits installés.
`Mode as TiaPortalMode`	Fournit en retour le mode dans lequel TIA Portal a été démarré (WithoutUserInterface/ WithUserInterface).
`AttachedSessions as Process[]`	Fournit en retour un tableau d'applications qui sont connectées à TIA Portal.
`ProjectPath as string`	Fournit en retour le nom de fichier du projet ouvert dans TIA Portal, y compris le dossier, par exemple. `"D:\WinCCProjects\ColorMixing\ColorMixing.ap14"` Si aucun projet n'est ouvert, une chaîne de caractères vide est fournie en retour.
`ID as string`	Fournit en retour l'ID de processus de l'instance TIA Portal.
`Path as string`	Fournit en retour le chemin d'accès au fichier exécutable de TIA Portal.

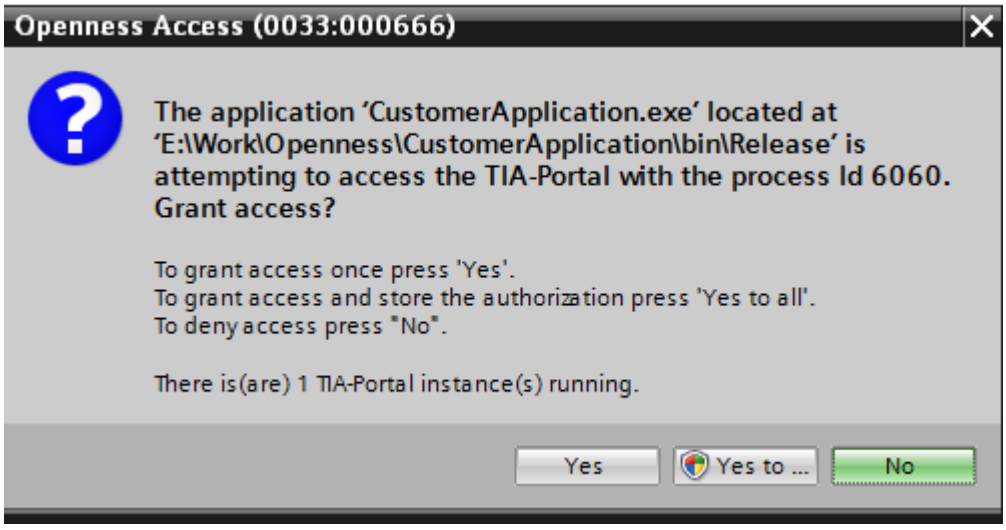
Voir aussi

Bibliothèques standard (Page 34)

7.9.3 Pare-feu Openness

Invite de commande du pare-feu Openness

Si vous essayez d'établir une liaison via Openness à une instance de TIA Portal en cours d'exécution, TIA Portal vous invite à accepter ou à rejeter la liaison, comme représenté sur la copie d'écran suivante.



Autoriser la liaison à TIA Portal de manière unique

Si vous souhaitez autoriser de manière unique la liaison de votre application Openness à TIA Portal, cliquez sur "Yes" à l'invite de commande. Lorsque votre application Openness essaiera d'établir une liaison à TIA Portal la prochaine fois, l'invite apparaîtra de nouveau.

Ajouter une entrée Whitelist par l'accès à TIA Portal

Pour créer une entrée Whitelist pour votre application Openness, procédez comme suit :

1. Cliquez sur "Yes to all" à l'invite. TIA Portal ouvre ensuite le contrôle de compte utilisateur.
2. Dans le contrôle de compte utilisateur, cliquez sur "Yes". Votre application est alors ajoutée à la Whitelist du pare-feu Openness et reliée à TIA Portal.

Ajouter une entrée Whitelist sans TIA Portal

Si vous voulez créer une entrée Whitelist sans utiliser TIA Portal, vous pouvez générer un fichier de Registre comme suit :

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\Whitelist
\CustomerApplication.exe]
[HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\Openness\14.0\Whitelist
\CustomerApplication.exe\Entry]
"Path"="E:\\Work\\Openness\\CustomerApplication\\bin\\Release\\CustomerApplication.exe"
"DateModified"="2014/06/10 15:09:44.406"
"FileHash"="0rXRKUCNzMWHOMFrT52OwXzqJef10ran4UykTeBraaY="
```

L'exemple suivant montre comment calculer la valeur Filehash et la date de dernière modification :

```
string applicationPath = @"E:\\Work\\Openness\\CustomerApplication\\bin\\Release\
\CustomerApplication.exe";
string lastWriteTimeUtcFormatted = String.Empty;
DateTime lastWriteTimeUtc;
HashAlgorithm hashAlgorithm = SHA256.Create();
FileStream stream = File.OpenRead(applicationPath);
byte[] hash = hashAlgorithm.ComputeHash(stream);
string convertedHash = Convert.ToBase64String(hash); // this is how the hash should appear
in the .reg file
lastWriteTimeUtc = fileInfo.LastWriteTimeUtc;
lastWriteTimeUtcFormatted = lastWriteTimeUtc.ToString(@"yyyy/MM/dd HH:mm:ss.fff"); // this
is how the last write time should be formatted
```

7.9.4 Gestionnaire d'événements

Gestionnaire d'événements dans l'application Openness

Une instance de TIA Portal propose les événements suivants, auxquels vous pouvez réagir avec un gestionnaire d'événements dans une application Openness. Vous pouvez accéder aux propriétés de notifications et définir les réactions en conséquence.

Evénement	Réponse
`Disposed`	Cet événement vous permet de réagir avec une application Openness à la fermeture de TIA Portal.
`Notification`	Cet événement vous permet de réagir avec une application Openness aux notifications de TIA Portal. Les notifications requièrent juste une confirmation, telle que "OK".
`Confirmation`	Cet événement vous permet de réagir avec une application Openness aux notifications de TIA Portal. Les confirmations requièrent toujours une décision, telle que "Voulez-vous enregistrer le projet ?".

Code du programme

Pour enregistrer un gestionnaire d'événements dans une application Openness, modifiez le code de programme suivant :

```
//Register event handler for Disposed-Event
    ....
        TiaPortal.Disposed +=TiaPortal_Disposed;
....

private static void TiaPortal_Disposed(object sender, EventArgs e)
{
    ....
}

//Register event handler for Notification-Event
    ....
        TiaPortal.Notification += TiaPortal_Notification;
....

private static void TiaPortal_Notification(object sender, NotificationEventArgs e)

{
    ....
}

//Register event handler for Confirmation-Event
    ....
        TiaPortal.Confirmation += TiaPortal_Confirmation;
....

private static void TiaPortal_Confirmation(object sender, ConfirmationEventArgs e)
{
    ....
}
```

Propriétés des notifications de TIA Portal

Les notifications de TIA Portal possèdent les propriétés suivantes :

Propriété	Description
`Caption`	Fournit en retour le nom de la confirmation.
`DetailText`	Fournit en retour le texte détaillé de la confirmation.
`Icon`	Fournit en retour l'icône de la confirmation.
`IsHandled`	Fournit en retour la confirmation ou indique si le système attend la confirmation.
`MessageID`	Fournit en retour l'ID univoque au sein du service.
`ServiceID`	Fournit en retour l'ID du service ayant déclenché la confirmation.
`Text`	Fournit en retour le texte de la confirmation.

Propriétés des confirmations

Les confirmations possèdent les propriétés suivantes :

Propriété	Description
`Caption`	Fournit en retour le nom de la confirmation.
`Choices`	Donne les possibilités pour acquitter la confirmation.
`DetailText`	Fournit en retour le texte détaillé de la confirmation.
`Icon`	Fournit en retour l'icône de la confirmation.
`IsHandled`	Fournit en retour la confirmation ou indique si le système attend la confirmation.
`MessageID`	Fournit en retour l'ID univoque au sein du service.
`Result`	Fournit ou indique le résultat de l'acquittement.
`ServiceID`	Fournit en retour l'ID du service ayant déclenché la confirmation.
`Text`	Fournit en retour le texte de la confirmation.

Voir aussi

Confirmer les boîtes de dialogue comportant des alarmes système par commande du programme (Page 77)

7.9.5 Confirmer les boîtes de dialogue comportant des alarmes système par commande du programme

Condition requise

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Les gestionnaires d'événements sont enregistrés.
  Voir Etablissement d'une connexion au portail TIA (Page 69)

Utilisation

Si vous commandez le portail TIA par le biais de l'interface utilisateur, des boîtes de dialogue comportant des événements système s'affichent pour certaines séquences du programme. À l'aide de ces événements système, vous décidez comment continuer.

Si vous accédez à TIA Portal avec une application Openness, ces événements système doivent être acquittés via les événements ".NET" correspondants.

Les confirmations autorisées figurent dans la liste `Choices` :

- `Abort`
- `Cancel`
- `Ignore`

- `No`
- `NoToAll`
- `None`
- `OK`
- `Retry`
- `Yes`
- `YesToAll`

La valeur de `ConfirmationEventArgs.Result` doit être l'une des entrées susmentionnées. Faute de quoi une exception est déclenchée.

Code du programme

Pour réagir à un événement de confirmation, modifiez le code de programme suivant :

```
private void TiaPortalOnConfirmation(object sender, ConfirmationEventArgs e)
{
    int serviceId = e.ServiceId;
    int messageId = e.MessageId;
    if (serviceId == MyExpectedServiceId)
    {
        if (messageId == MyExpectedMessageId && ((e.Choices & ConfirmationChoices.Yes) ==
ConfirmationChoices.Yes))
        {
            e.Result = ConfirmationResult.Yes;
            e.IsHandled = true;
        }
    }
}
```

Pour informer le concepteur des actions exécutées avec une application Openness, modifiez le code de programme suivant :

```
//Handles notifications
using (TiaPortal tiaPortal = new TiaPortal())
{
    tiaPortal.Notification += Notification;
    try
        {
            //perform actions that will result in a notification event
        }
    finally
        {
            tiaPortal.Notification -= Notification;
        }
}
```

```
private void TiaPortalOnNotification(object sender, NotificationEventArgs e)
{
    if (e.MessageId == MyExpectedMessageId && e.ServiceId == MyExpectedServiceId)
    {
        e.IsHandled = true;
        }
    }
```

Voir aussi

Bibliothèques standard (Page 34)

7.9.6 Mettre fin à la connexion au portail TIA

Introduction

Si vous avez démarré l'instance du portail TIA sans interface utilisateur, vous pouvez fermer cette instance du portail TIA via l'application Openness. Dans tous les autres cas, déconnectez l'application Openness de l'instance du portail TIA. Si un concepteur ferme l'instance du portail TIA en dépit d'un accès en cours d'une application Openness, une exception de la catégorie "NonRecoverableException" est déclenchée dans l'application Openness.

Déconnectez ou fermez l'instance active de TIA Portal à l'aide de la méthode `IDisposable.Dispose()`.

Vous pouvez utiliser la méthode `IDisposable.Dispose()` comme suit :

- avec un using-Statement.
- Entourez la description de l'objet avec un bloc try-finally et appelez la méthode `IDispose.Dispose()` au sein du bloc finally.

Si vous quittez l'instance active du portail TIA, vous ne pouvez plus accéder au portail TIA.

Code du programme

Pour couper la connexion au portail TIA ou y mettre fin, utilisez le code de programme suivant :

```
// Add code to dispose the application if the application is still instantiated
if (tiaPortal != null)
{
    tiaPortal.Dispose();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.9.7 Interfaces de diagnostic dans TIA Portal

Utilisation

Vous pouvez appeler certaines informations de diagnostic d'instances de TIA Portal en cours d'exécution avec une méthode statique. L'interface de diagnostic est réalisée dans l'objet TiaPortalProcess que vous pouvez appeler pour chaque instance TIA Portal en cours d'exécution.

L'interface de diagnostic n'est pas bloquée. Vous pouvez donc accéder à l'objet TiaPortalProcess ainsi qu'à ses membres et ce, que TIA Portal soit occupé ou non. L'interface de diagnostic comprend les membres suivants :

Classe TiaPortalProcess

Membre	Type	Fonction
AcquisitionTime	DateTime	Date et heure auxquelles l'objet TiaPortalProcess a été saisi. Comme l'objet TiaPortalProcess représente un instantané tout à fait statique de l'état de TIA Portal à un moment donné, les informations qu'il contient peuvent être obsolètes.
Attach	TiaPortal	Est relié au TiaPortalProcess donné et fournit en retour une instance de TIA Portal.
AttachedSessions	IList<TiaPortalSession>	Bibliothèque de toutes les autres sessions actuellement attachées au même TIA Portal. Cette bibliothèque peut être vide. Chaque session est représentée par un objet TiaPortalSession décrit comme suit.
Attaching	EventHandler<AttachingEventArgs>	Cet événement permet à l'application d'autoriser des tentatives de lien à TIA Portal. Lorsqu'une autre application essaie de s'attacher à TIA Portal, les membres de cet événement en sont informés. Les membres ont alors 10 secondes pour autoriser l'opération. Si un membre ignore cet événement ou n'y réagit pas à temps, cela est considéré comme un refus et l'opération n'est pas autorisée à l'application demandeuse. Les applications bloquées, et donc incapables de réagir à cet événement, ne peuvent pas refuser à une autre application l'autorisation de s'attacher.

Membre	Type	Fonction
Dispose	void	Ferme l'instance de TIA Portal correspondante.
Id	int	ID de processus de TIA Portal.
InstalledSoftware	IList<TiaPortalProduct>	Bibliothèque de tous les produits actuels installés comme partie de TIA Portal. Chaque produit est représenté par un objet TiaPortalProduct décrit comme suit.
Mode	TiaPortalMode	Fournit en retour le mode dans lequel TIA Portal a été démarré. Valeurs actuelles : WithUserInterface et WithoutUserInterface.
Path	string	Chemin d'accès au fichier exécutable de TIA Portal.
ProjectPath	string	Chemin d'accès au projet actuel ouvert dans TIA Portal. Si aucun projet n'est ouvert, cette propriété a la valeur zéro.

Classe TiaPortalSession

Membre	Type	Fonction
AccessLevel	TiaPortalAccessLevel	Niveau d'accès de la session. Est représenté sous forme d'une énumération de mémentos et plusieurs niveaux d'accès sont possibles. La section qui suit propose une description détaillée de TiaPortalAccessLevel.
AttachTime	DateTime	Date et heure auxquelles la liaison à TIA Portal été établie.
Id	int	L'ID de la session actuelle.
IsActive	bool	Fournit "vrai" en retour lorsque le traitement d'un appel provenant de cette session est en cours d'exécution sur TIA Portal.
Dispose	void	Met fin à la connexion entre le processus et TIA Portal. Cette méthode ne force pas la fin du processus, comme ce serait le cas avec System.Diagnostics.Process.Kill. L'application dont la connexion est terminée reçoit malgré tout un événement Disposed. Toutefois, aucune indication n'est donnée sur ce qui a provoqué la fin de la connexion.
ProcessId	int	ID du processus attaché.
ProcessPath	string	Chemin d'accès au fichier exécutable du processus attaché.

Membre	Type	Fonction
TrustAuthority	TiaPortalTrustAuthority	Indique si la session actuelle a été démarrée par un processus signé et s'il s'agit d'un certificat Openness ou non. TrustAuthority est une énumération de mémentos décrite comme suit.
UtilizationTime	TimeSpan	Période pendant laquelle le processus a été actif dans TIA Portal. En combinaison avec la propriété AttachTime, ceci peut servir à déterminer des pourcentages de durée de vie ou d'autres informations similaires.
Version	string	Version de Siemens.Engineering.dll attaché à la session.

Enum TiaPortalAccessLevel

Valeur d'énumération	Fonction
None	Cette valeur n'est pas valide. La valeur est indiquée parce que TiaPortalAccessLevel est un mémento de type Enum et qu'à ce titre, une "valeur zéro" correspondante est requise pour montrer qu'aucun mémento n'est mis à 1. Mais elle n'apparaît jamais dans la pratique, car aucune session ne peut être démarrée sans accès.
Published	La session a accès à la fonctionnalité publiée.
Prepublished	La session a accès à la fonctionnalité prépubliée.
Modify	La session a accès en modification.

Enum TiaPortalTrustAuthority

Valeur d'énumération	Fonction
None	Le module principal du processus attaché n'est pas signé à l'aide d'un certificat.
CustomerIdentification	Le module principal utilise un fichier CustomerID.
Signed	Le module principal est signé à l'aide d'un certificat, mais il ne s'agit pas d'un certificat Openness.
Certified	Le module principal est signé à l'aide d'un certificat.

Classe TiaPortalProduct

Membre	Type	Fonction
Name	string	Nom du produit (par ex. STEP 7 Professional).
Options	IList<TiaPortalProduct>	Une bibliothèque de tous les progiciels qui font partie du TIA Portal connecté, représentés comme objets TiaPortalProduct. Lorsqu'un progiciel contient, à son tour, des progiciels, le niveau d'imbrication peut être plus important.
Version	string	Trame de version du produit.

L'exemple de section de code suivant montre comment utiliser l'interface de diagnostic pour interroger des informations et comment ensuite utiliser ces informations dans votre application.

```
public void TiaPortalDiagnostics()
{
    IList<TiaPortalProcess> tiaPortalProcesses = TiaPortal.GetProcesses();
    foreach (TiaPortalProcess tiaPortalProcess in tiaPortalProcesses)
    {
        Console.WriteLine("Process ID: {0}", tiaPortalProcess.Id);
        Console.WriteLine("Path: {0}", tiaPortalProcess.Path);
        Console.WriteLine("Project: {0}", tiaPortalProcess.ProjectPath);
        Console.WriteLine("Timestamp: {0}", tiaPortalProcess.AcquisitionTime);
        Console.WriteLine("UI Mode: {0}", tiaPortalProcess.Mode);
        //See method body below.
        Console.WriteLine("Installed Software:");
        EnumerateInstalledProducts(tiaPortalProcess.InstalledSoftware);
        Console.WriteLine("Attached Openness Applications:");
        foreach (TiaPortalSession session in tiaPortalProcess.AttachedSessions)
        {
            Console.WriteLine("Process: {0}", session.ProcessPath);
            Console.WriteLine("Process ID: {0}", session.ProcessId);
            DateTime attachTime = session.AttachTime;
            TimeSpan timeSpentAttached = DateTime.Now - attachTime;
            TimeSpan utilizationTime = session.UtilizationTime;
            long percentageTimeUsed = (utilizationTime.Ticks / timeSpentAttached.Ticks) *
100;
            Console.WriteLine("AttachTime: {0}", attachTime);
            Console.WriteLine("Utilization Time: {0}", utilizationTime);
            Console.WriteLine("Time spent attached: {0}", timeSpentAttached);
            Console.WriteLine("Percentage of attached time spent using TIA Portal: {0}",
percentageTimeUsed);
            Console.WriteLine("AccessLevel: {0}", session.AccessLevel);
            Console.WriteLine("TrustAuthority: {0}", session.TrustAuthority);
            if ((session.TrustAuthority & TiaPortalTrustAuthority.Certified) !=
TiaPortalTrustAuthority.Certified)
            {
                Console.WriteLine("TrustAuthority doesn't match required level, attempting
to terminate connection to TIA Portal."); session.Dispose();
            }
        }
    }
}
public void EnumerateInstalledProducts(IEnumerable<TiaPortalProduct> products)
{
    foreach (TiaPortalProduct product in products)
    {
        Console.WriteLine("Name: {0}", product.Name);
        Console.WriteLine("Version: {0}", product.Version);
        //recursively enumerate all option packages
        Console.WriteLine("Option Packages \n:");
        EnumerateInstalledProducts(product.Options);
    }
}
```

Information de sécurité

Du fait que l'utilisation de l'interface de diagnostic ne nécessite aucune liaison à TIA Portal, il est possible d'écrire un service Windows qui se sert de l'événement latéral pour vérifier toute application qui essaie de s'attacher à TIA Portal. Ainsi, il est possible par ex. de définir que seules les applications commençant par le nom de votre entreprise sont autorisées à s'attacher. Une autre option pourrait consister à autoriser systématiquement l'accès mais à écrire les informations sur les processus latéraux dans un journal. Le code du programme suivant donne un exemple de gestionnaire d'événements pour vérifier des liaisons entrantes :

```
public void OnAttaching(object sender, AttachingEventArgs e)
{
    string name = Path.GetFileNameWithoutExtension(e.ProcessPath);
    TiaPortalAccessLevel requestedAccessLevel = e.AccessLevel &
TiaPortalAccessLevel.Published;
    TiaPortalTrustAuthoritycertificateStatus = e.TrustAuthority
&TiaPortalTrustAuthority.Certified;
    if (requestedAccessLevel == TiaPortalAccessLevel.Published &&
       certificateStatus == TiaPortalTrustAuthority.Certified &&
       name.StartsWith("SampleCustomerName"))
    {
        e.GrantAccess();
    }
}
```

7.9.8 Exclusive access

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Pour une application Client Openness qui a besoin d'un accès exclusif à un processus de TIA Portal attaché, la classe "TIA Portal" propose la méthode "ExclusiveAccess(Stringtext)" pour configurer un processus exclusif.

Remarque

Toute tentative visant à créer un deuxième accès exclusif dans l'étendue d'un accès exclusif ouvert, entraîne le déclenchement d'une exception qui peut être restaurée.

Remarque

Utilisez "ExclusiveAccess" dans une instruction "using" pour vous assurer que l'accès est correctement terminé lorsque des exceptions se produisent ou l'application est arrêtée.

Modifiez l'exemple suivant pour avoir "ExclusiveAccess" à une instance :

```
...
[assembly: AssemblyTitle("MyApplication")]
// Est utilisé pour la boîte de dialogue d'accès exclusif, si présent...
TiaPortal tiaPortal = ...;
using (ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess("My Activity"))
{
    ...
}
```

Après la saisie d'une instance 'ExclusiveAccess' pour un processus de TIA Portal donné, une boîte de dialogue s'ouvre. Cette boîte de dialogue affiche le message prévu lors de l'instanciation. En plus, les informations suivantes sur l'application Client sont également affichées :

- le titre d'Assembly des données manifestes, s'il est disponible, ou le nom du processus
- l'ID du processus
- le SID

Remarque

Plusieurs sessions peuvent être actives pour une application Client Openness donnée parce qu'il existe plusieurs instances de TIA Portal, même si celles-ci sont attribuées respectivement au même processus TIA Portal.

L'application Client peut également actualiser le contenu affiché dans la boîte de dialogue d'accès exclusif en remplaçant les valeurs de la propriété "Texte" par de nouvelles valeurs. Pour obtenir ce comportement, modifiez le code de programme suivant :

```
exclusiveAccess = ...;
...
exclusiveAccess.Text = "My Activity Phase 1";
...
exclusiveAccess.Text = "My Activity Phase 2";
...
exclusiveAccess.Text = String.Empty; // or null;
...
```

Vous pouvez exiger l'annulation de l'accès exclusif à l'aide du bouton "Cancel"/"Annuler". Pour obtenir ce comportement, modifiez le code de programme suivant :

```
exclusiveAccess = ...;
...
if (exclusiveAccess.IsCancellationRequested)
{
    // Arrêtez votre action
    ...
}
else
{
    // Poursuivez votre action
    ...
}
...
```

7.9.9 Traitement des transactions

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Opération

Une rémanence (projet, bibliothèque, etc.) ouverte au sein d'un processus TIA Portal correspondant peut être modifiée par une application Client Openness. Vous pouvez obtenir cette modification avec une opération unique ou une série d'opérations. Pendant l'action, il est utile de regrouper ces opérations dans une seule pile Undo afin d'obtenir un déroulement plus logique. En outre, regrouper des opérations dans une pile Undo unique présente également des avantages en matière de performance. Pour aider à cela, la classe "ExclusiveAccess" propose la méthode "Transaction(ITransactionSupport persistence, string undoDescription)". L'appel depuis cette méthode entraîne l'instanciation d'un nouvel objet de type "Transaction" qui peut être arrêté. Vous devez fournir la description du contenu de la transaction (l'attribut Texte ne doit pas être nul ou vide). Tant que cette instance n'a pas été terminée, toutes les opérations des applications Client sont regroupées dans une pile Undo unique au sein du processus TIA Portal correspondant.

Pour saisir une instance de type "Transaction", modifiez le code de programme suivant :

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation"))
{
    ...
}
```

Remarque

Utilisez une instruction "using" pour instancier une "Transaction". Cela vous permet d'assurer que la transaction soit terminée correctement, même si des exceptions se produisent, et d'annuler ainsi la transaction.

Application cohérente ou annulation

Une "Transaction" dans une application Client vous permet d'assurer qu'il existe un chemin prévisible pour rendre effectif ou annuler un jeu de modifications. Votre application Client doit décider si les modifications apportées à une rémanence doivent devenir effectives ou pas. Pour cela, votre application doit demander que les modifications dans l'étendue d'une transaction ouvertes deviennent effectives lorsque la transaction est terminée par appel de la méthode "Transaction.CommitOnDispose()". Si cette méthode n'est jamais appelée dans le déroulement du code, les modifications apportées dans l'étendue de la transaction ouverte sont automatiquement annulées à la fin de la transaction.

Si une exception se produit après cette requête, toutes les modifications apportées dans l'étendue d'une transaction ouverte sont malgré tout annulées à la fin de la transaction.

Pour créer une pile Undo unique dans un processus TIA Portal attaché avec deux modifications "Create", modifiez le code de programme suivant :

```
ExclusiveAccess exclusiveAccess = ...;
Project project = ...;
using (Transaction transaction = exclusiveAccess.Transaction(project, "My Operation")
{
    project.DeviceGroups.Create("My Group 1");
    project.DeviceGroups.Create("My Group 2");
    transaction.CommitOnDispose();
}
```

Comportement Undo

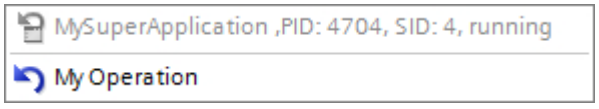
Les actions exécutées par une application Client Openness peuvent résulter en piles Undo à l'intérieur du processus TIA Portal attaché. Chacune de ces entrées Undo est regroupée sous une entrée locale. L'entrée locale est composée des informations suivantes provenant de l'application Client :

- le titre d'Assembly des données manifestes, s'il est disponible, ou le nom du processus
- l'ID du processus

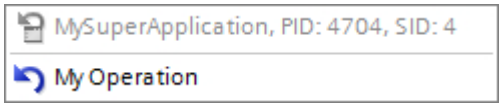
- le SID ou
- un message indiquant que le processus Client est en cours d'exécution

Ces entrées appartiennent à un des deux types suivants :

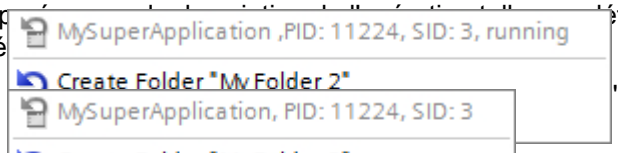
1. Les opérations, qui sont regroupées dans une transaction Undo comme résultats de l'utilisation d'une "Transaction", possèdent la description mise à disposition par l'application Client lorsque la "Transaction" a été instanciée.
   – Entrée Undo pour une application Client en cours d'exécution :

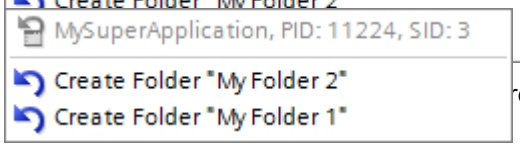
   – Entrée Undo pour une application Client arrêtée :

2. Pour les opérations qui sont exécutées individuellement, il existe des entrées Undo sép finie dans la commande de mé
   – 'exécution :

   – êtée :

7.10 Fonctions des projets/données de projet

7.10.1 Ouvrir un projet

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)

Remarque

Le projet à ouvrir ne doit être ouvert dans aucune autre instance du portail TIA.

Remarque

Aucun accès aux projets en écriture seule

TIA Portal Openness V14 ne peut accéder qu'aux projets avec droits en lecture et en écriture.

Utilisation

Utilisez la méthode `Projects.Open` pour ouvrir les projets qui ont été créés avec TIA Portal V 14 ou qui ont été mis à niveau vers la version 14.

Placez la commande d'ouverture du projet dans un bloc try et la commande de fermeture dans un bloc finally.

Dans la méthode `Projects.Open`, entrez un chemin pour le projet de votre choix.

Remarque

Accès aux projets de TIA Portal V13 SP1

La méthode `Projects.Open` ne peut accéder qu'aux projets qui ont été créés avec TIA Portal V14 ou qui ont été mis à niveau vers la version 14.

Si vous accédez à un projet d'une version précédente avec la méthode `Projects.Open` , vous obtiendrez une exception en retour.

Utilisez la méthode `OpenWithUpgrade` pour ouvrir les projets qui ont été créés avec une version précédente de TIA Portal.

Code du programme

Pour ouvrir un projet, modifiez le code de programme suivant :

```
Project project =tiaPortal.Projects.Open(@"D:\Some\Path\Here\Project.apXX");
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Ouvrir un projet qui a été créé avec une version précédente

Utilisez la méthode `OpenWithUpgrade` pour ouvrir un projet qui a été créé avec une version précédente de TIA Portal. Cette méthode permet de mettre à niveau le projet vers la version actuelle et de l'ouvrir.

Code du programme

Pour ouvrir un projet avec la méthode OpenWithUpgrade, modifiez le code de programme suivant :

```
Project project = tiaPortal.Projects.OpenWithUpgrade(@"D:\Some\Path\Here
\Project.apXX");
if (project != null)
{
    try
    {
        ...
    }
    finally
    {
        project.Close();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.2 Enumérer et appeler des appareils

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Application : Accès à un appareil

Pour appeler des objets de type "IDeviceItem", utilisez la propriété suivante :

- Nom (string) : nom de l'appareil

Créez le code de votre programme conformément à l'exemple suivant :

```
IDevice device = ...
string name = device.Name;
```

Application : Énumération d'appareils

Remarque

Faire attention à Hiérarchie des objets matériels du modèle d'objet (Page 59).

Utilisez une des possibilités suivantes pour énumérer les appareils d'un projet :

- Énumérer tous les appareils du premier niveau
- Énumérer tous les appareils en groupes ou sous-groupes
- Énumérer tous les appareils d'un projet ne contenant aucun groupe d'appareils
- Énumérer tous les appareils des groupes de système d'appareils non groupés

Exemples d'appareils pouvant être énumérés :

- Central station
- PB-Slave / PN-IO device
- HMI Device

Code du programme : énumérer les appareils du premier niveau

Pour énumérer les appareils du premier niveau, utilisez le code de programme suivant :

```
private static void EnumerateDevicesInProject(Project project)
{
    IDeviceComposition deviceComposition = project.Devices;
    foreach (IDevice device in deviceComposition)
    {
    // add code here
    }
}
```

Pour accéder à un appareil en particulier, modifiez le code de programme suivant :

```
private static void AccessSingleDeviceByName(Project project)
{
    IDeviceComposition deviceComposition = project.Devices;
    // The parameter specifies the name of the device
    Device device = deviceComposition.Find("MyDevice")as Device;
}
```

Code du programme : Énumérer des appareils en groupes ou sous-groupes

Remarque

Avant de pouvoir accéder aux appareils d'un groupe, vous devez d'abord naviguer jusqu'au groupe, ensuite jusqu'à l'appareil.

Pour énumérer les appareils en groupes et sous-groupes, modifiez le code de programme suivant :

```
private static void EnumerateDevicesInGroups(Project project)
{
    foreach (DeviceUserGroup deviceUserGroup in project.DeviceGroups)
    {
        EnumerateDeviceUserGroup(deviceUserGroup);
    }
}
private static void EnumerateDeviceUserGroup(DeviceUserGroup deviceUserGroup)
{
    EnumerateDeviceObjects(deviceUserGroup.Devices);
    foreach (deviceUserGroup subDeviceUserGroup in deviceUserGroup.Groups)
    {
        // recursion
        EnumerateDeviceUserGroup(subDeviceUserGroup);
    }
}
private static void EnumerateDeviceObjects(IDeviceComposition deviceComposition)
{
    foreach (IDevice device in deviceComposition)
    {
    // add code here
    }
}
```

Code du programme : Énumérer les appareils d'un projet ne contenant aucun groupe d'appareils

Pour énumérer tous les appareils qui se trouvent sous un projet ne contenant aucun groupe d'appareils, modifiez le code de programme suivant :

```
Project project = ...
foreach (IDevice device in project.Devices)
{
    ... // Work with the devices
}
```

Pour rechercher un appareil en particulier avec son nom, modifiez le code de programme suivant :

```
Project project = ...
IDevice plc1 = project.Devices.FirstOrDefault(d => d.Name == "Mydevice");
... // Work with the device
```

Pour rechercher un appareil en particulier avec la méthode "Find", modifiez le code de programme suivant :

```
Project project = ...
IDevice plc1 = project.Devices.Find("MyDevice");
... // Work with the device
```

Code du programme : Énumérer les appareils des groupes de système d'appareils non groupés

Remarque

Pour structurer les projets, des appareils décentralisés ont été ajoutés dans le groupe UngroupedDevices. Avant de pouvoir accéder au groupe correspondant, vous devez d'abord naviguer jusqu'au groupe, ensuite jusqu'à l'appareil.

Pour énumérer tous les appareils du groupe UngoupedDevices, modifiez le code de programme suivant :

```
Project project = ...
DeviceSystemGroup group = project.UngroupedDevicesGroup;
IDevice plc1 = group.Devices.First(d => d.Name == "MyStationName");
... // Work with the device
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.3 Enumérer et appeler des éléments d'appareils

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Application : Appeler des éléments d'appareils

Pour appeler des objets de type "IDeviceItem", utilisez les propriétés suivantes :

- Nom (string) : Nom de l'élément d'appareil
- Conteneur (IHardwareObject) : Conteneur dans lequel l'élément d'appareil est placé

Code du programme : Appeler l'élément d'appareil

Pour appeler un élément d'appareil, modifiez le code de programme suivant :

```
public static IDeviceItem AccessDeviceItemFromDevice(Device device)
{
    IDeviceItem deviceItem = device.DeviceItems[0];
    return deviceItem;
}
```

Code du programme : Accéder à un élément d'appareil depuis un élément d'appareil

Pour accéder à un élément d'appareil depuis un élément d'appareil, modifiez le code de programme suivant :

```
public static IDeviceItem AccessDeviceItemFromDeviceItem(DeviceItem deviceItem)
{
    IDeviceItem subDeviceItem = deviceItem.DeviceItems[0];
    return subDeviceItem;
}
```

Application : Énumérer des éléments d'appareils

Utilisez "Items" sur IHardwareObject. pour énumérer les éléments d'appareils : Il peut s'agir des éléments suivants :

- châssis dans un appareil
- module dans un châssis
- sous-module dans un module

Remarque

Avant de pouvoir accéder aux éléments d'appareil internes à un niveau inférieur, Il est possible que vous soyez obligé de naviguer au travers plusieurs niveaux d'éléments d'appareil.

Vous trouverez plus d'informations à ce sujet au chapitre Hiérarchie des objets matériels du modèle d'objet (Page 59).

Code du programme : Énumérer les éléments d'un appareil

Pour énumérer les éléments d'appareil d'un objet matériel, modifiez le code de programme suivant :

```
private static void EnumerateDeviceItems(IHardwareObject hardwareObject)
{
    foreach (IDeviceItem deviceItem in hardwareObject.Items)
    {
        // add code here
    }
}
```

Code du programme : Énumérer au moyen de la hiérarchie de composition

Pour énumérer les éléments d'un appareil au moyen de la hiérarchie de composition, modifiez le code de programme suivant :

```
//Enumerates devices using an composition
private static void EnumerateDeviceItems(Device device)
{
    IDeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (IDeviceItem deviceItem in deviceItemComposition)
    {
        // add code here
    }
}
```

Code du programme : Énumérer des éléments d'appareils avec affectation

Pour énumérer des éléments d'appareils au moyen d'une affectation, modifiez le code de programme suivant :

```
//Enumerates devices using an association
private static void EnumerateDeviceItemsWithAssociation(Device device)
{
    IDeviceItemAssociation deviceItemAssociation = device.Items;
    foreach (IDeviceItem deviceItem in deviceItemAssociation)
    {
        // add code here
    }
}
```

Code du programme : Naviguer jusqu'au conteneur d'un élément d'appareil

Pour retourner au conteneur d'un élément d'appareil modifiez le code de programme suivant avec la propriété ""Container"" de IDeviceItem :

```
IDeviceItem deviceItem = ...;
IHardwareObject container = deviceItem.Container;
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.4 Déterminer la structure et les attributs de l'objet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez déterminer la structure de navigation de la hiérarchie d'objet à l'aide de l'interface `IEngineeringObject`. Le résultat est retourné sous forme de liste :

- Objets inférieurs
- Compositions inférieures
- Tous les attributs aux droits d'accès spécifiques, comme la lecture ou l'écriture.

Signature

Utilisez la méthode `GetAttributeInfos` pour déterminer les attributs aux droits d'accès spécifiques.

```
IList<EngineeringAttributeInfo>
IEngineeringObject.GetAttributeInfos(AttributeAccessMode
attributeAccessMode);
```

Paramètres	Fonction
`AttributeAccessMode`	Définit le droit d'accès avec lequel les attributs de l'objet indiqué sont interrogés. • `Read` : fournit en retour tous les attributs avec l'accès "Lecture". • `ReadOnly` : fournit en retour tous les attributs avec l'accès "Lecture seule". • `Write` : fournit en retour tous les attributs avec l'accès "Ecriture". • `WriteOnly` : fournit en retour tous les attributs avec l'accès "Ecriture seule". • `ReadWrite` : fournit en retour tous les attributs avec l'accès "Lecture et écriture".

Code de programme : Déterminer des objets ou compositions

Si vous connaissez la valeur de retour, modifiez le code de programme suivant :

```
public static void DisplayCompositionInfos(Device device)
{
    IList<EngineeringCompositionInfo> compositionInfos =
((IEngineeringObject)device).GetCompositionInfos();
    foreach (EngineeringCompositionInfo compositionInfo in compositionInfos)
    {
        Console.WriteLine(compositionInfo.Name);
    }
}
```

Si vous ne connaissez pas la valeur de retour, modifiez le code de programme suivant :

```
public static IDeviceItemComposition GetDeviceItemComposition(Device device)
{
    IEngineeringCompositionOrObject composition = ((IEngineeringObject)
device).GetComposition("DeviceItems");
    IDeviceItemComposition deviceItemComposition = (IDeviceItemComposition)composition;
    return deviceItemComposition;
}
```

Code de programme : détermination des attributs

Pour fournir en retour les attributs d'un objet aux droits d'accès spécifiques dans une liste, modifiez le code de programme suivant :

```
public static void DisplayAttributenInfos(Device device)
{
    IList<EngineeringAttributeInfo> attributeInfos =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo attributeInfo in attributeInfos)
    {
        Console.WriteLine("Attribute: {0} - AccessMode {1} ",
        attributeInfo.Name, attributeInfo.AccessMode);
        switch (attributeInfo.AccessMode)
        {
            case EngineeringAttributeAccessMode.Read: Console.WriteLine("Attribute: {0} -
Read Access", attributeInfo.Name);
            break;
            case EngineeringAttributeAccessMode.Write: Console.WriteLine("Attribute: {0} -
Write Access", attributeInfo.Name);
            break;
          case EngineeringAttributeAccessMode.Read | EngineeringAttributeAccessMode.Write:
Console.WriteLine("Attribute: {0} - Read and Write Access", attributeInfo.Name);
            break;
        }
    }
}
public static string GetDeviceNameAttribute(Device device)
{
    Object nameAttribute = ((IEngineeringObject)device).GetAttribute("Name");
    string name = (string)nameAttribute; return name;
}
```

7.10.5 Attributs obligatoires d'appareils et d'éléments d'appareils

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Chaque appareil ou élément d'appareil possède certains attributs obligatoires pouvant être lus et/ou écrits. Ces attributs sont toujours identiques à ceux de l'interface utilisateur de TIA Portal.

Openness prend en charge les attributs suivants :

Nom d'attribut	Type de données	Accessible en écriture	Description
Nom	Chaîne de caractères	read	
PositionNumber	int	read	Uniquement pour éléments d'appareils

Code du programme : Attributs obligatoires d'un appareil

Pour appeler les attributs obligatoires d'un appareil, modifiez le code de programme suivant :

```
IDevice device = ...;
string nameValue = device.Name;
```

Code du programme : Attributs obligatoires d'un élément d'appareil

Pour appeler les attributs obligatoires d'un élément d'appareil, modifiez le code de programme suivant :

```
IDeviceItem deviceItem = ...;
string nameValue = deviceItem.Name;
int positionNumberValue = deviceItem.PositionNumber;
```

7.10.6 Ouvrir l'éditeur "Appareils & réseaux"

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez ouvrir l'éditeur "Appareils & réseaux" avec l'interface API grâce à l'une des deux méthodes suivantes :

- `ShowHwEditor(View.Topology` ou `Network` ou `Device)` : Ouvre l'éditeur "Appareils & Réseaux" depuis le projet.
- `ShowInEditor(View.Topology` ou `Network` ou `Device)` : Affiche l'appareil indiqué dans l'éditeur "Appareils & Réseaux".

Le paramètre `View` vous permet de définir la vue qui est affichée lors de l'ouverture de l'éditeur :

- `View.Topology`
- `View.Network`
- `View.Device`

Code du programme

Pour ouvrir l'éditeur "Appareils & Réseaux", modifiez le code de programme suivant :

```
// Open topology view from project
private static void OpenEditorDevicesAndNetworksFromProject(Project project)
{
    project.ShowHwEditor(Siemens.Engineering.HW.View.Topology);
}
```

Pour ouvrir l'éditeur "Appareils & Réseaux" pour un appareil, modifiez le code de programme suivant :

```
// Open topology view for given device
private static void OpenEditorDevicesAndNetworksFromDevice(Device device)
{
    device.ShowInEditor(Siemens.Engineering.HW.View.Topology);
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.10.7 Interroger PLC Target et HMI Target

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez définir si un logiciel de base peut être utilisé comme PLC Target (PlcSoftware) ou HMI Target dans l'API public.

Code du programme : PLC Target

Pour vérifier si un élément d'appareil peut être utilisé comme PLC Target, utilisez le code de programme suivant :

```
// Returns PlcSoftware
private PlcSoftware GetPlcSoftware(Device device)
{
    IDeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        ISoftwareContainer softwareContainer = deviceItem.GetService<ISoftwareContainer>();
        if (softwareContainer != null)
        {
            SoftwareBase softwareBase = softwareContainer.Software;
            PlcSoftware plcSoftware = softwareBase as PlcSoftware;
            return plcSoftware;
        }
    }
    return null;
 }
```

Code du programme : HMI Target

Pour vérifier si un élément d'appareil peut être utilisé comme HMI Target, modifiez le code de programme suivant :

```
//Checks whether a device is of type hmitarget
private HmiTarget GetHmiTarget(Device device)
{
    IDeviceItemComposition deviceItemComposition = device.DeviceItems;
    foreach (DeviceItem deviceItem in deviceItemComposition)
    {
        ISoftwareContainer softwareContainer = deviceItem.GetService<ISoftwareContainer>();
        if (softwareContainer != null)
        {
            SoftwareBase softwareBase = softwareContainer.Software;
            HmiTarget hmiTarget = softwareBase as HmiTarget;
            return hmiTarget;
        }
    }
    return null;
}
```

Voir aussi

Bibliothèques standard (Page 34)

Enumérer et appeler des appareils (Page 92)

7.10.8 Accéder au logiciel cible

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour mettre à disposition un logiciel cible, modifiez le code de programme suivant :

```
ISoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<ISoftwareContainer>();
if (softwareContainer != null)
{
    SoftwareBase softwareBase = softwareContainer.Software;
}
```

Pour accéder aux propriétés du logiciel, modifiez le code de programme suivant :

```
ISoftwareContainer softwareContainer =
((IEngineeringServiceProvider)deviceItem).GetService<ISoftwareContainer>();
if (softwareContainer != null)
{
    PlcSoftware software = softwareContainer.Software as PlcSoftware;
    string name = software.Name;
}
```

7.10.9 Interroger un groupe système pour variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Tous les appareils sont énumérés.
  Voir Enumérer et appeler des appareils (Page 92)
- Tous les éléments d'appareil d'un appareil API sélectionné sont énumérés.
  Voir Enumérer et appeler des éléments d'appareils (Page 95)
- Tous les éléments d'appareils contenant des tables de variables sont décelés.
  Voir Interroger PLC Target et HMI Target (Page 102)

Code du programme

Pour interroger un groupe système pour variables API, modifiez le code de programme suivant :

```
//Retrieves the plc tag table group from a plc
private PlcTagTableSystemGroup GetControllerTagfolder(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    return plcTagTableSystemGroup;
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.10 Enumérer les groupes personnalisés pour variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Tous les appareils sont énumérés.
  Voir Enumérer et appeler des appareils (Page 92)
- Tous les éléments d'appareil d'un appareil API sélectionné sont énumérés.
  Voir Enumérer et appeler des éléments d'appareils (Page 95)
- Tous les éléments d'appareils contenant des tables de variables sont décelés.
  Voir Interroger PLC Target et HMI Target (Page 102)

Utilisation

Les sous-dossiers compris sont considérés comme récurrents lors de l'énumération.

Code du programme : Enumérer les groupes personnalisés pour variables API

Pour énumérer les groupes personnalisés pour variables API, modifiez le code de programme suivant :

```
//Enumerates all plc tag table user groups including subgroups
private static void EnumeratePlcTagTableUserGroups(PlcSoftware plcSoftware)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in plcSoftware.TagTableGroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
    }
}
private static void EnumerateTagTableUserGroups(PlcTagTableUserGroup tagTableUsergroup)
{
    foreach (PlcTagTableUserGroup plcTagTableUsergroup in tagTableUsergroup.Groups)
    {
        EnumerateTagTableUserGroups(plcTagTableUsergroup);
        // recursion
    }
}
```

Code du programme : Accéder à un groupe personnalisé

Pour accéder à un groupe personnalisé pour variables API, modifiez le code de programme suivant :

```
//Gives individual access to a specific plc tag table user folder
private static void AccessPlcTagTableUserGroupWithFind(PlcSoftware plcSoftware, string
folderToFind)
{
    PlcTagTableUserGroupComposition plcTagTableUserGroupComposition =
plcSoftware.TagTableGroup.Groups;
    PlcTagTableUserGroup controllerTagUserFolder =
plcTagTableUserGroupComposition.Find(folderToFind);
    // The parameter specifies the name of the user folder
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.11 Enumérer des variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme : Enumérer des variables API dans des tables de variables

Pour énumérer toutes les variables API contenues dans une table des variables, modifiez le code de programme suivant :

```
//Enumerates all plc tags in a specific tag table
private static void EnumerateAllPlcTagsInTagTable(PlcSoftware plcSoftware)
{
    PlcTagTable tagTable = plcSoftware.TagTableGroup.TagTables.Find("Tagtable XYZ");
    foreach (PlcTag tag in tagTable.Tags)
    {
        // add code here
    }
}
```

Code du programme : Accéder à une variable API

Pour accéder à la variable API de votre choix, modifiez le code de programme suivant : Vous avez accès aux propriétés suivantes :

- Nom
- Type de données
- Adresse symbolique
- Commentaire

```
//Gives individual access to a specific plc tag
private static void AccessPlcTag(PlcTagTable tagTable)
{
    PlcTag tag = tagTable.Tags.Find("Tag XYZ");
    // The parameter specifies the name of the tag
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.12 Enumérer des tables de variables API dans un dossier

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme : Enumérer des tables de variables API

Pour énumérer toutes les tables de variables API contenues dans les groupes système ou groupes personnalisés, modifiez le code de programme suivant :

```
//Enumerates all plc tag tables in a specific system group or and user group
private static void EnumerateAllPlcTagTablesInFolder(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    foreach (PlcTagTable tagTable in tagTables)
    {
        // add code here
    }
}
```

Code du programme : Accéder à une table de variables API

Pour accéder à la table de variables API de votre choix, modifiez le code de programme suivant :

```
//Gives individual access to a specific Plc tag table
private static void AccessToPlcTagTableWithFind(PlcSoftware plcSoftware)
{
    PlcTagTableComposition tagTables = plcSoftware.TagTableGroup.TagTables;
    // alternatively, PlcTagTableComposition tagTables =
plcSoftware.TagTableGroup.Groups.Find("UserGroup XYZ").TagTables;
    PlcTagTable controllerTagTable = tagTables.Find("Tag table XYZ");
    // The parameter specifies the name of the tag table
}
```

Voir aussi

Bibliothèques standard (Page 34)

Enumérer et appeler des appareils (Page 92)

7.10.13 Interroger les informations d'une table de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez connaître le nombre de variables d'une table de variables API grâce au numéro de composition des variables.

Code du programme

Pour interroger les informations d'une table de variables API, modifiez le code de programme suivant :

```
private static void AccessPlcConstantsUsingFind(PlcTagTable tagTable)
{
    PlcConstantComposition plcConstants = tagTable.Constants;
    PlcConstant plcConstant = plcConstants.Find("Constant XYZ");
}
private static void EnumeratePlcTags(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    foreach (PlcTag plcTag in plcTags)
    {
        string name = plcTag.Name; string typeName = plcTag.DataTypeName;
        string logicalAddress = plcTag.LogicalAddress;
    }
}
private static void EnumeratePlcTagsUsingFind(PlcTagTable tagTable)
{
    PlcTagComposition plcTags = tagTable.Tags;
    PlcTag plcTag = plcTags.Find("Constant XYZ");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.14 Compiler le projet

Condition

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Tous les appareils sont "hors ligne".

Application

L'interface API prend en charge la compilation d'appareils et de blocs de programme. Le résultat de la compilation est retourné en tant qu'objet. Selon le type d'objet, la compilation HW, SW ou HW/SW est mise à disposition. Les types d'objet suivants sont pris en charge :

- `Device - HW & SW`
- `DeviceItem - HW`
- `PlcSoftware - SW`
- `CodeBlock - SW`
- `DataBlock - SW`
- `PlcType - SW`
- `PlcBlockSystemGroup - SW`
- `PlcBlockUserGroup - SW`
- `PlcTypeSystemGroup - SW`
- `PlcTypeUserGroup - SW`

Signature

Pour la compilation, utilisez la méthode `ICompilable`.

```
ICompilable compileService = <eom object>.GetService<ICompilable>();
CompilerResult = compileService.Compile();
```

Remarque

Pour compiler le matériel d'un appareil, utilisez le type d'objet `Device`.

Remarque

Tous les appareils doivent être "hors ligne" avant le début de la compilation.

Code du programme

Pour compiler les modifications logicielles d'un objet de type `PlcSoftware`, modifiez le code de programme suivant :

```
public static void CompilePlcSoftware(PlcSoftware plcSoftware)
{
    ICompilable compileService = plcSoftware.GetService<ICompilable>();
    CompilerResult result = compileService.Compile();
}
```

Pour compiler les modifications logicielles d'un objet de type `CodeBlock`, modifiez le code de programme suivant :

```
public static void CompileCodeBlock(PlcSoftware plcSoftware)
{
    CodeBlock block = plcSoftware.BlockGroup.Blocks.Find("MyCodeBlock") as CodeBlock;
    if (block != null)
    {
        ICompilable compileService = block.GetService<ICompilable>();
        CompilerResult result = compileService.Compile();
    }
}
```

Pour évaluer le résultat de la compilation, modifiez le code de programme suivant :

```
private void WriteCompilerResults(CompilerResult result)
{
    Console.WriteLine("State:" + result.State);
    Console.WriteLine("Warning Count:" + result.WarningCount);
    Console.WriteLine("Error Count:" + result.ErrorCount);
    RecursivelyWriteMessages(result.Messages);
}
private void RecursivelyWriteMessages(CompilerResultMessageComposition messages, string
indent = "")
{
    indent += "\t";
    foreach (CompilerResultMessage message in messages)
    {
        Console.WriteLine(indent + "Path: " + message.Path);
        Console.WriteLine(indent + "DateTime: " + message.DateTime);
        Console.WriteLine(indent + "State: " + message.State);
        Console.WriteLine(indent + "Description: " + message.Description);
        Console.WriteLine(indent + "Warning Count: " + message.WarningCount);
        Console.WriteLine(indent + "Error Count: " + message.ErrorCount);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.10.15 Fonctions pour bibliothèques

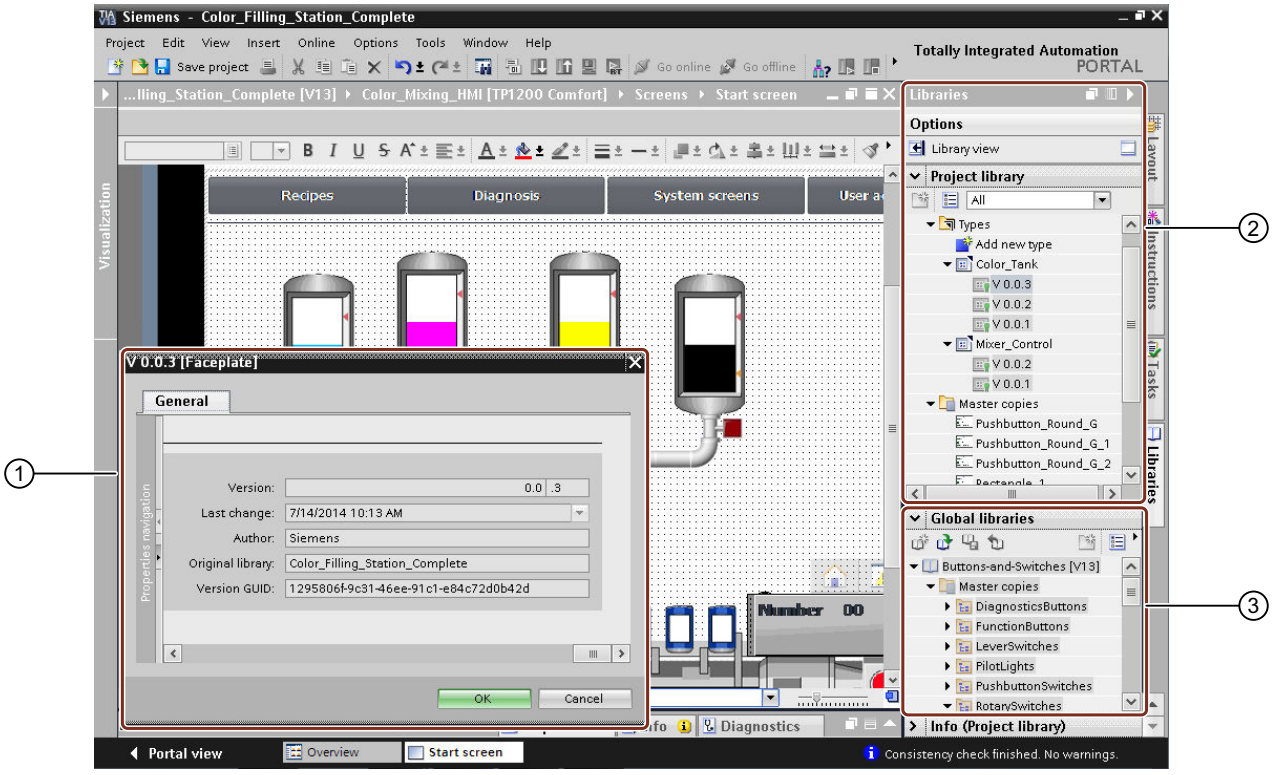
7.10.15.1 Fonctions pour objets et instances

Accès aux types et instances

L'interface Public API vous permet d'accéder aux types, versions de types et copies maîtresse dans la bibliothèque de projet ou les bibliothèques globales. Vous pouvez déterminer des liaisons entre les versions de types et les instances. Vous pouvez également actualiser les instances dans le projet et synchroniser les modifications entre une bibliothèque globale et la bibliothèque de projet. En outre, l'interface Public API prend en charge la comparaison des versions de type et des instances.

Fonctions pour des objets et des instances

L'interface Public API vous permet d'accéder aux fonctions suivantes pour les types, versions de types, copies maîtresse et instances :



① Afficher les propriétés de types, versions de types, copies maîtresse et instances

② Les fonctions suivantes sont disponibles dans la bibliothèque de projet :

- Mettre à jour les instances des types
- Instancier les versions de types dans le projet
- Naviguer à l'intérieur d'un groupe de bibliothèques
- Supprimer un groupe, des types, des versions de types et des copies maîtresses

③ Les fonctions suivantes sont disponibles dans la bibliothèque globale :

- Mettre à jour les instances des types
- Instancier la version de type dans le projet
- Naviguer à l'intérieur d'un groupe de bibliothèques

7.10.15.2 Accéder aux bibliothèques

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface Public API vous permet d'accéder aux contenus suivants de la bibliothèque globale et de la bibliothèque de projet.

Code du programme

Remarque

Vous pouvez uniquement accéder à des bibliothèques globales à partir de TIA Portal V14 avec Openness. Si vous ouvrez des bibliothèques globales d'une version antérieure, une exception est déclenchée.

Remarque

`GlobalLibraries.Open` n'ouvre pas la bibliothèque globale dans l'interface utilisateur de TIA Portal.

`GlobalLibraries` ne fournit pas de liste des bibliothèques ouvertes en retour.

Pour accéder à une bibliothèque globale, modifiez le code de programme suivant :

```
public static void AccessGlobalLibrary(TiaPortal tiaPortal)
{
    String strPathToGlobalLib = @"C:\OpennessSamples\MyGlobalLib\MyGlobalLib.al14";
    GlobalLibrary globalLibrary = tiaPortal.GlobalLibraries.Open(strPathToGlobalLib);

}
```

Pour accéder à la bibliothèque de projet, modifiez le code de programme suivant :

```
public static void AccessProjectLibrary(Project project)
{
    ProjectLibrary projectLibrary = project.ProjectLibrary;
}
```

Voir aussi

Accéder aux dossiers dans une bibliothèque (Page 115)

7.10.15.3 Accéder aux dossiers dans une bibliothèque

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).

Utilisation

L'interface Public API vous permet d'accéder aux dossiers système pour les types et copies maître dans une bibliothèque. Vous pouvez alors accéder aux types, versions de types, copies maître et dossiers personnalisés dans le dossier système.

La méthode `Find`, par ex. `libTypeUserFolder.Folders.Find("SomeUserFolder");`, vous permet d'accéder à tout moment à un dossier personnalisé.

Code du programme

Pour accéder au dossier système pour les types dans une bibliothèque, modifiez le code de programme suivant :

```
public static void AccessTypeSystemFolder(ILibrary library)
{
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
}
```

Pour accéder au dossier système pour les copies maître dans une bibliothèque, modifiez le code de programme suivant :

```
public static void AccessMasterCopySystemFolder(ILibrary library)
{
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
}
```

Pour accéder à des dossiers personnalisés avec la méthode `Find()` , modifiez le code de programme suivant :

```
...
LibraryTypeUserFolderComposition userFolderComposition = ...
LibraryTypeUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Pour énumérer les dossiers personnalisés dans un dossier système, modifiez le code de programme suivant :

```
public static void EnumerateUserFoldersInTypeSystemFolder(ILibrary library)
{
    // Enumerating user folders in type system folder:
    LibraryTypeSystemFolder libTypeSystemFolder = library.TypeFolder;
    foreach (LibraryTypeUserFolder libTypeUserFolder in libTypeSystemFolder.Folders)
    {
        //...
    }
}
```

```
public static void EnumerateUserFoldersInMasterCopySystemFolder(ILibrary library)
{
    // Enumerating user folders in master copy system folder:
    MasterCopySystemFolder libMasterCopySystemFolder = library.MasterCopyFolder;
    foreach (MasterCopyUserFolder libMasterCopyUserFolder in
libMasterCopySystemFolder.Folders)
    {
        //..
    }
}
```

Pour énumérer les sous-dossiers personnalisés dans un dossier personnalisé pour des types, modifiez le code de programme suivant :

```
public static void EnumerateAllUserFolders(LibraryTypeUserFolder libUserFolder)
{
    foreach (LibraryTypeUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Pour énumérer les sous-dossiers personnalisés dans un dossier personnalisé pour des copies maître, modifiez le code de programme suivant :

```
public static void EnumerateAllUserFolders(MasterCopyUserFolder libUserFolder)
{
    foreach (MasterCopyUserFolder libSubUserFolder in libUserFolder.Folders)
    {
       EnumerateAllUserFolders(libSubUserFolder);
    }
}
```

Voir aussi

Accéder à des modèles de copie (Page 125)

7.10.15.4 Accéder aux types

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un groupe pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

Vous pouvez accéder aux types d'une bibliothèque par le biais d'une interface Public API.

- Vous pouvez énumérer les types.
- Vous pouvez accéder aux propriétés suivantes des différents types :

Propriété	type de données	Description
Author	String	Fournit le nom de l'auteur :
Comment	MultilingualText	Fournit en retour le commentaire.
Guid	Guid	Fournit en retour le GUID du type.[1]
Name	String	Fournit en retour le nom du type. [2]

[1] Cette propriété vous permet de trouver un type précis dans une bibliothèque. La recherche est récursive.

[2] Cette propriété vous permet de trouver un type précis dans un dossier. Les sous-dossiers ne sont pas pris en compte dans la recherche.

Code du programme

Pour énumérer tous les types dans le dossier système d'une bibliothèque, modifiez le code de programme suivant :

```
public static void EnumerateTypesInTypesSystemFolder(LibraryTypeSystemFolder
libraryTypeSystemFolder)
{
    foreach (LibraryType libraryType in libraryTypeSystemFolder.Types)
    {
        //...
    }
}
```

Pour énumérer tous les types dans le dossier personnalisé d'une bibliothèque, modifiez le code de programme suivant :

```
public static void EnumerateTypesInTypesUserFolder (LibraryTypeUserFolder
libraryTypeUserGroup)
{
    foreach (LibraryType libraryType in libraryTypeUserGroup.Types)
    {
        //...
    }
}
```

Pour accéder aux propriétés d'un type, modifiez le code de programme suivant :

```
public static void InspectPropertiesOfType (LibraryType libTypeObject)
{
    string typeAuthor = libTypeObject.Author;
    MultilingualText typeComment = libTypeObject.Comment;
    string typeName = libTypeObject.Name;
    Guid typeGUID = libTypeObject.Guid;
}
```

Pour trouver un type précis par son nom ou son GUID, modifiez le code de programme suivant :

```
public static void FindTypeObjectInLibrary(ILibrary library)
{
    // Find type object by its GUID in a given library:
    System.Guid targetGuid = ...;
    LibraryType libTypeByGUID = library.FindType(targetGuid);
    // Find type object by its name in a given group:
    ILibraryTypeFolder libTypeSystemFolder = library.TypeFolder;
    LibraryType libTypeByName = libTypeSystemFolder.Types.Find("myTypeObject");
}
```

7.10.15.5 Accéder aux types de versions

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un groupe pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

Vous pouvez accéder aux versions de type par le biais de l'interface Public API.

- Vous pouvez énumérer les versions de types d'un type.
- Vous pouvez déterminer le type auquel cette version de type appartient.
- Vous pouvez déterminer les instances d'une version de type.
- Vous pouvez déterminer les copies maîtresses contenant des instances.
- Vous pouvez déterminer les instances d'une version de type.
- Vous pouvez créer une nouvelle instance d'une version de type.
- Vous pouvez accéder aux propriétés suivantes des différentes versions de type :

Propriété	type de données	Description
Author	String	Fournit le nom de l'auteur :
Comment	MultilingualText	Fournit en retour le commentaire.
Guid	Guid	Fournit en retour le GUID de la version de type.[1]
ModifiedDate	DateTime	Fournit en retour la date et l'heure à laquelle la version de type a été mise au statut "`Committed`".
State	LibraryTypeVersionState	Fournit en retour l'état de la version : • `InWork` : Correspond, selon le type affecté, au statut "En cours" ou "Test en cours". • `Committed` : Correspond au statut "Validé".
TypeObject	LibraryType	Fournit en retour les types auxquels cette version de type appartient.
VersionNumber	Version	Fournit en retour le numéro de version sous forme de désignation à trois chiffres, p. ex. "1.0.0".[2]

[1] Cette propriété vous permet de trouver une version de type précise dans une bibliothèque.

[2] Cette propriété vous permet de trouver une version de type précise dans une composition "LibraryTypeVersion".

Déterminer les utilisations d'une version de type

Les utilisations suivantes sont différenciées pour les versions de type :

- La version de type utilise d'autres versions de types issues de la bibliothèque.
  Exemple : un type de données personnalisé est utilisé dans un bloc de programme. Le bloc de programme doit avoir accès au type de données utilisateur. Cela signifie que le bloc de programme dépend du type de données utilisateur.
  Si vous appelez la méthode `GetDependencies()` dans le bloc de programme, le type de données utilisateur est fourni en retour.
- Le type est utilisé par une autre version de type dans la bibliothèque.
  Exemple : un type de données personnalisé est utilisé dans un bloc de programme. Le bloc de programme doit avoir accès au type de données utilisateur. Le type de données utilisateur a le bloc de programme correspondant. Le bloc de programme dépend du type de données utilisateur.
  Si vous appelez la méthode `GetDependents()` dans le type de données utilisateur, le bloc de programme est fourni en retour.

Dans les deux méthodes, une liste contenant des objets du type `LibraryTypeVersion` est fournie en retour. Si aucune utilisation n'existe, une chaîne vide est fournie en retour.

Remarque

Une exception peut être déclenchée si vous appliquez cette méthode aux versions de type avec le statut "`InWork`".

Déterminez à la place l'instance test du type d'objet `ILibraryTypeInstance` de cette version de type et appliquez les méthodes à cet objet.

Déterminer les instances d'une version de type

Vous pouvez déterminer les instances d'une version de type par la méthode `FindInstancesInProject(IInstanceSearchScope searchScope).`

Le paramètre `searchScope` vous permet d'indiquer le champ de recherche au sein du projet. Les classes suivantes implémentent l'interface `IInstanceSearchScope` et peuvent être utilisées pour la recherche d'instances :

- `PlcSoftware`
- `HmiTarget`

Cette méthode fournit en retour une liste comprenant des objets du type `ILibraryTypeInstance`. Si aucune instance n'existe, une chaîne vide est fournie en retour.

Créer l'instance d'une version de type

Vous pouvez créer une nouvelle instance d'une version de type. Les objets suivants sont pris en charge :

- Blocs (FB/FC)
- Types de données personnalisé API
- Vues
- Scripts VB

Une instance permet de générer une version de type d'une bibliothèque globale et de la bibliothèque de projet. Si vous créez une instance d'une version de type depuis une bibliothèque globale, la version de type est d'abord synchronisée avec la bibliothèque de projet.

Signature

Utilisez la méthode `Instantiate` pour créer une instance.

```
ILibraryTypeInstance Instantiate(ILibraryTypeInstantiationTarget
instantiationTarget, UpdatePathsMode updatePathsMode)
```

Paramètres	Fonction
`ILibraryTypeInstantiationTarget instantiationTarget`	Indique le dossier où l'instance est créée. Le groupe système et le groupe personnalisé qu'ils contient sont pris en charge en relation avec les objets indiqués ci-dessus.
`UpdatePathsMode updatePathsMode`	Définit la réaction lorsqu'une instance de cette version de type est déjà disponible dans le groupe cible : • `UpdatePathsInTarget` : L'instance existante est déplacée dans le groupe indiqué. • `KeepExistingPathsInTarget` : L'instance existante est conservée dans le groupe d'origine. • `ThrowIfPathsConfict` : une exception est déclenchée. L'exécution est annulée. Aucune instance n'est créée.

Code du programme

Pour énumérer toutes les versions de type d'un type, modifiez le code de programme suivant :

```
public static void EnumerateVersionsInType(LibraryType libraryType)
{
    foreach (LibraryTypeVersion libraryTypeVersion in libraryType.Versions)
    {
        //...
    }
}
```

Pour déterminer le type auquel appartient une version de type, modifiez le code de programme suivant :

```
public static void GetParentTypeOfVersion(LibraryTypeVersion libTypeVersion)
{
    LibraryType parentType = libTypeVersion.TypeObject;
}
```

Pour déterminer les utilisations d'une version de type dans une bibliothèque, modifiez le code de programme suivant :

```
public static void GetDependenciesAndDependentsOfAVersion(LibraryTypeVersion
libTypeVersion)
{
    IList<LibraryTypeVersion> versionDependents = libTypeVersion.GetDependents();
    IList<LibraryTypeVersion> versionDependencies = libTypeVersion.GetDependencies();
}
```

Pour déterminer les instances d'une version de type dans le projet, modifiez le code de programme suivant :

```
public static void GetInstancesOfVersionInProject(LibraryTypeVersion libTypeVersion,
PlcSoftware plcSoftware)
{
    IInstanceSearchScope searchScope = plcSoftware as IInstanceSearchScope; // or HmiTarget
    if (searchScope != null)
    {
        IList<ILibraryTypeInstance> instances =
libTypeVersion.FindInstancesInProject(searchScope);
    }
}
```

Pour déterminer les copies maîtresses contenant des instances d'une version de type, modifiez le code de programme suivant :

```
public static void GetMasterCopiesContainingInstances(LibraryTypeVersion libTypeVersion)
{
    MasterCopyAssociation masterCopies = libTypeVersion.MasterCopiesContainingInstances;
}
```

Pour créer une instance d'une version de type, modifiez le code de programme suivant :

```
public static void InstantiateTypeVersion(LibraryTypeVersion libTypeVersion,
ILibraryTypeInstantiationTarget instantiationTarget)
{
    ILibraryTypeInstance instance = libTypeVersion.Instantiate(instantiationTarget,
UpdatePathsMode.UpdatePathsInTarget);
}
```

Pour accéder aux propriétés d'une version de type, modifiez le code de programme suivant :

```
public static void InspectPropertiesOfVersion(LibraryTypeVersion libTypeVersion)
{
    string versionAuthor = libTypeVersion.Author;
    MultilingualText versionComment = libTypeVersion.Comment;
    Guid versionGUID = libTypeVersion.Guid; DateTime versionModifiedDate =
libTypeVersion.ModifiedDate;
    LibraryTypeVersionState versionStateLibrary = libTypeVersion.State;
    LibraryType versionParentObject = libTypeVersion.TypeObject;
    Version versionNumber = libTypeVersion.VersionNumber;
}
```

Pour trouver une version de type précise par son numéro de version, modifiez le code de programme suivant :

```
public static void FindVersionInLibrary(ILibrary library, Guid versionGUID)
{
    LibraryTypeVersion libTypeVersionByVersionNumber = library.FindVersion(versionGUID);
}
```

7.10.15.6 Accéder aux instances

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un groupe pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

Vous pouvez accéder aux instances des versions de type par le biais de l'interface Public API.

Remarque

Les instances des blocs d'affichage et des types de données utilisateur HMI sont toujours associées à la version de type correspondante.

Les instances de tous les autres objets tels que les blocs de programme ou les vues peuvent être associées à une version de type.

Code du programme

Les classes suivantes peuvent implémenter une ILibraryTypeInstance :

- Script VB - IHM
- Vue - IHM
- Variable - IHM
- FB - SW
- FC - SW
- Structure - SW

Pour accéder à une instance, modifiez le code de programme suivant :

```
public static void GetInstance()
{
    FB codeBlock = null; // Or screen, datatype, etc. Browsed from the project
    ILibraryTypeInstance libTypeInstance = codeBlock as ILibraryTypeInstance;
}
```

Si, par ex. une instance d'un type de données utilisateur API est utilisé dans l'instance d'une interface de bloc, les deux instances dépendent l'une de l'autre.

Pour déterminer les instances dépendant d'une instance, modifiez le code de programme suivant :

```
public static void GetDependents(ILibraryTypeInstance libTypeInstance)
{
    ILibraryTypeInstanceAssociation dependentInstances = libTypeInstance.Dependents;
}
```

Pour déterminer l'instance dépendant d'une instance, modifiez le code de programme suivant :

```
public static void GetDependencies(ILibraryTypeInstance libTypeInstance)
{
    ILibraryTypeInstanceAssociation dependencyInstances = libTypeInstance.Dependencies;
}
```

Pour déterminer la version de type associée à une instance, modifiez le code de programme suivant : Avant de poursuivre, vérifiez que l'instance est associée à une version de type.

```
public static void NavigateFromInstanceToObjectVersion(ILibraryTypeInstance
libTypeInstance)
{
    LibraryTypeVersion connectedVersion = libTypeInstance.ConnectedVersion;
    if (connectedVersion != null)
    {
        //instance object is connected to a library type-version
        //...
    }
    else
    {
        //instance object is not connected to a library type-version
        //...
    }
}
```

Pour énumérer les dépendances d'une instance de type de bibliothèque, modifiez le code de programme suivant :

```
public static void EnumerateDependenciesOfLibraryTypeInstance(ILibraryTypeInstance
libTypeInstance, FB fb)
{
    ILibraryTypeInstance instance = fb as ILibraryTypeInstance;
    ILibraryTypeInstanceAssociation dependencyInstances = instance.Dependencies;
}
```

7.10.15.7 Accéder à des modèles de copie

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113)
- Vous avez accès à un groupe pour les copies maîtresses.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115)

Utilisation

L'interface Public API prend en charge l'accès aux copies maîtresse dans une bibliothèque globale et la bibliothèque de projet :

- Enumérer des copies maîtresses dans des dossiers système et des dossiers personnalisés
- Interroger les informations des copies maîtresses

Propriété	type de données	Description
Author	String	Fournit le nom de l'auteur :
CreationDate	DateTime	Fournit en retour la date de création.
Name	String	Fournit en retour le nom de la copie maîtresse.

Code du programme

Pour énumérer toutes les copies maîtresse dans le dossier système d'une bibliothèque, modifiez le code de programme suivant :

```
public static void EnumerateMasterCopiesInSystemFolder
(MasterCopySystemFolder masterCopySystemFolder)
{
    foreach (MasterCopy masterCopy in masterCopySystemFolder.MasterCopies)
    {
        //...
    }
}
```

Pour accéder à une copie maîtresse individuelle par la méthode "Find", modifiez le code de programme suivant :

```
...
    MasterCopySystemFolder systemFolder = projectLibrary.MasterCopyFolder;
    MasterCopyComposition mastercopies = systemFolder.MasterCopies;
    MasterCopy masterCopy = mastercopies.Find("Copy of ...");
...
```

Pour énumérer les groupes et sous-groupes des copies maîtresses, modifiez le code de programme suivant :

```
foreach (MasterCopyUserFolder userFolder in systemFolder.Folders)
{
    EnumerateUserFolder(userFolder) ...
}
private static void EnumerateUserFolder(MasterCopyUserFolder userFolder)
{
    EnumerateMasterCopies(userFolder.MasterCopies);
    foreach (MasterCopyUserFolder subUserFolder inuserFolder.Folders)
    {
        EnumerateUserFolder(subUserFolder); // recursion
    }
}
private static void EnumerateMasterCopies(MasterCopyComposition masterCopies)
{
    foreach (MasterCopy masterCopy in masterCopies)
    {
        ...
    }
}
```

Pour accéder à un MasterCopyUserFolder par la méthode "Find", modifiez le code de programme suivant :

```
...
    MasterCopyUserFolderComposition userFolderComposition = ...
    MasterCopyUserFolder userFolder = userFolderComposition.Find("Name of user folder");
...
```

Pour lire les informations d'une copie maîtresse, modifiez le code de programme suivant :

```
public static void GetMasterCopyInformation(MasterCopy masterCopy)
{
    string author = masterCopy.Author;
    DateTime creationDate = masterCopy.CreationDate;
    string name = masterCopy.Name;
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

Copier un modèle de copie (Page 132)

Supprimer les contenus de bibliothèque (Page 142)

Copier le contenu d'un modèle de copie dans le projet (Page 128)

7.10.15.8 Créer la copie maîtresse d'un projet dans la bibliothèque

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Lorsqu'il s'agit d'une bibliothèque en lecture/écriture, vous pouvez créer une copie maîtresse issue d'une IMasterCopySource sur l'emplacement cible.

Pour créer la copie maîtresse sur l'emplacement cible, modifiez le code de programme suivant :

```
MasterCopy
MasterCopyComposition.Create(Siemens.Engineering.Library.MasterCopies.IMasterCopySource
sourceObject);
```

Une exception EngineeringException est déclenchée dans les cas suivants :

- L'emplacement cible est accessible en lecture seule
- Le système refuse la génération d'une copie maîtresse issue de la source

Les éléments suivants sont définis comme IMasterCopySources :

- Device - HW
- DeviceItem - HW
- DeviceUserGroup - HW
- CodeBlock - SW
- DataBlock - SW
- PlcBlockUserGroup - SW
- PlcTag - SW
- PlcTagTable - SW
- PlcTagTableUserGroup - SW

- PlcType - SW
- PlcTypeUserGroup - SW
- VBScript - HMI
- VBScriptUserFolder - HMI
- Screen - HMI
- ScreenTemplate - HMI
- ScreenTemplateUserFolder - HMI
- ScreenUserFolder - HMI
- Tag - HMI
- TagTable - HMI
- TagUserFolder - HMI

Code du programme

Pour créer une copie maîtresse issue d'une bibliothèque de projets, modifiez le code de programme suivant :

```
public static void Create(Project project, PlcSoftware plcSoftware)
{
    MasterCopySystemFolder masterCopyFolder = project.ProjectLibrary.MasterCopyFolder;
   CodeBlock block = plcSoftware.BlockGroup.Groups[0].Blocks.Find("Block_1") as CodeBlock;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Create(block);
}
```

7.10.15.9 Copier le contenu d'un modèle de copie dans le projet

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

L'interface Public API prend en charge l'utilisation de copies maîtresse dans le projet. Vous pouvez copier le contenu d'une copie maîtresse dans un dossier du projet ouvert.

Si vous copiez le contenu d'une copie maîtresses, le système fournit en retour une liste des copies maîtresse copiées comme résultat. Les objets maître suivants sont pris en charge :

- Groupes d'utilisateurs d'appareils
- Appareils
- Eléments d'appareils
- Vues
- Modèles
- Scripts
- Tables de variables IHM
- Type de données IHM
- Groupes d'utilisateurs de bloc API
- Variable API
- Tables de variables API
- Groupe d'utilisateurs de table de variables API
- Groupe d'utilisateurs de type API
- Types de données API
- Blocs de programme
  - Blocs d'organisation (OB)
  - Fonctions (FC)
  - Blocs fonctionnels (FB)
  - Blocs de données (DB)

  Exemple : Vous copiez une copie maîtresse qui contient une table de variables avec des variables. La table de variables et les variables sont copiées. La liste fournie en retour ne contient que la table des variables.

Signature

Pour copier le contenu d'une copie maîtresse, utilisez la méthode `CopyTo` :

```
IList<IEngineeringObject> copiedItems CopyTo(IMasterCopyTarget
target, MasterCopyMode copyMode)
```

Paramètre	Fonction
`IMasterCopyTarget target`	Objet du dossier cible dans la navigation de projet dans lequel le contenu de la copie maîtresse est copié. Si le contenu et le dossier cible ne concordent pas, une exception est déclenchée. Exemple : La copie maîtresse contient des blocs de programme. Vous indiquez `ScreenUserFolder` comme dossier cible.
`MasterCopyMode`	Le mode de copie définit la procédure suivie lorsqu'un objet de même nom est déjà disponible dans le lieu de stockage cible. • `ThrowIfExists` Le processus de copie est annulé et une exception est déclenchée. L'exception contient les informations sur l'objet ayant déclenché l'exception. • `Rename` L'objet à copier est renommé selon les règles de nom de TIA Portal et collé au lieu de stockage cible. • `Replace` Les objets de même nom se trouvant au lieu de stockage cible sont remplacés par les objets copiés.

Code du programme

Pour copier le contenu d'une copie maîtresse d'une bibliothèque vers la navigation du projet, modifiez le code de programme suivant :

```
public static void CopyMasterCopyToFolder(MasterCopy masterCopy, PlcBlockUserGroup
plcBlockUserGroup)
{
    IList<IEngineeringObject> copiedObjects = masterCopy.CopyTo(plcBlockUserGroup,
MasterCopyMode.ThrowIfExists);
}
```

Pour copier une copie maîtresse d'une bibliothèque de projets dans un projet, modifiez le code de programme suivant :

```
public static void CopyMasterCopyToUserFolder(Project project, PlcSoftware plcSoftware)
{
    MasterCopySystemFolder masterCopyFolder = project.ProjectLibrary.MasterCopyFolder;
    CodeBlock block = plcSoftware.BlockGroup.Groups[0].Blocks.Find("Block_1") as
CodeBlock;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Create(block);
}
```

Pour copier une copie maîtresse issue d'une bibliothèque globale dans un projet, modifiez le code de programme suivant :

```
public static void CopyMasterCopyToGlobalLibrary(GlobalLibrary globalLibrary,
PlcBlockUserGroup plcBlockUserGroup)
{
    MasterCopySystemFolder masterCopyFolder = globalLibrary.MasterCopyFolder;
    MasterCopy masterCopy = masterCopyFolder.MasterCopies.Find("Copy of Block_1");
    IList<IEngineeringObject> copiedObjects = masterCopy.CopyTo(plcBlockUserGroup,
MasterCopyMode.ThrowIfExists);
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

Accéder à des modèles de copie (Page 125)

7.10.15.10 Copier un objet copie maîtresse issu d'une bibliothèque globale dans la bibliothèque de projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez copier des objets copies maîtresses au sein d'une bibliothèque et entre bibliothèques en appliquant l'action `MasterCopy CopyTo(IMasterCopyFolder targetFolder)` sur les objets Siemens.Engineering.Library.MasterCopies.MasterCopy .

L'action CopyTo essaie de créer une copie de la copie maîtresse source dans le dossier cible.

- Si cette action se termine avec succès, la nouvelle copie maîtresse est fournie en retour.
- En cas de conflit pendant la copie dans un dossier cible, une copie renommée de la copie maîtresse est enregistrée dans le dossier cible. Le changement de nom est géré par le système grâce à un comportement de changement de nom par défaut.

IMasterCopyFolder peut être un MasterCopySystemFolder ou un MasterCopyUserFolder. Vous pouvez utiliser le dossier d'origine de la copie maîtresse comme dossier cible. Si tel est le cas, une copie supplémentaire est enregistrée dans le dossier.

Remarque

Les bibliothèques globales ne peuvent pas servir de destination de copie. Si le paramètre dossier cible est une bibliothèque globale, une exception est déclenchée.

Code du programme

Pour copier un objet copie maîtresse issue d'une bibliothèque globale dans une bibliothèque de projet, modifiez le code de programme suivant :

```
ProjectLibrary projectLib = ...;
MasterCopy globalLibMC = ...;
MasterCopy copyInProjectLibrary = globalLibMC.CopyTo(projectLib.MasterCopyFolder);
```

7.10.15.11 Copier un modèle de copie

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface Public API prend en charge la copie de copies maîtresse :

- Vous pouvez copier une copie maîtresse issue d'une bibliothèque globale dans la bibliothèque de projet.
- Vous pouvez copier une copie maîtresse dans la bibliothèque de projet.

Si vous copiez une copie maîtresse, le système fournit en retour la copie maîtresse copiée comme résultat.

Signature

Vous pouvez copier une copie maîtresse issue d'une bibliothèque globale dans la bibliothèque de projet à l'aide de la méthode `CopyTo`.

```
MasterCopy CopyTo(IMasterCopyFolder targetFolder)
```

Paramètres	Fonction
`IMasterCopyFolder targetFolder`	Indique le dossier où la copie maîtresse est copiée. Le dossier système ou un dossier personnalisé s'y trouvant peut être utilisé comme dossier. S'il existe une copie maîtresse portant le même nom, une copie de la copie maîtresse est ajoutée.

Vous trouverez une liste des cibles disponibles sous Créer la copie maîtresse d'un projet dans la bibliothèque (Page 127).

Code du programme

Pour copier une copie maîtresse issue d'une bibliothèque globale dans le dossier système des copies maîtresses dans la bibliothèque de projet, modifiez le code de programme suivant :

```
public static void CopyMasterCopyToFolder(MasterCopy masterCopyFromGlobalLib,
ProjectLibrary projectLibrary)
{
    MasterCopy copyInProjectLib =
masterCopyFromGlobalLib.CopyTo(projectLibrary.MasterCopyFolder);
}
```

Pour copier une copie maîtresse dans la bibliothèque de projet, modifiez le code de programme suivant :

```
public static void CopyMasterCopyToUserFolder(MasterCopy masterCopyFromProjectLib,
MasterCopyUserFolder userFolder)
{
    MasterCopy copyInProjectLibFolder = masterCopyFromProjectLib.CopyTo(userFolder);
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

Accéder à des modèles de copie (Page 125)

7.10.15.12 Déterminer les versions de types d'instances

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113)
- Vous avez accès à un dossier pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

L'interface Public API prend en charge la détermination des versions de types faisant partie des instances dans le projet ouvert. L'interface Public API fournit un des deux états suivants pour chaque instance :

- L'instance se rapporte à une version de type obsolète.
- L'instance se rapporte à la version de type actuelle.

Les règles applicables pour la détermination de version sont les suivantes :

- La détermination de version est basée sur une bibliothèque et le projet que vous souhaitez ouvrir via l'interface Public API.
- Aucune instance n'est actualisée dans le cadre de la détermination de version.

Signature

Vous pouvez déterminer les instances d'une version de type par la méthode `UpdateCheck` :
`UpdateCheck(Project project, UpdateCheckMode updateCheckMode)`

Paramètres	Fonction
`Project`	Définit le projet dans lequel les versions de types des instances sont déterminées. Cela est uniquement requis si vous déterminez des versions de types d'une bibliothèque globale.
`UpdateCheckMode`	Indique les versions qui sont déterminées : • `ReportOutOfDateOnly` : fournit en retour seulement l'état de type "obsolète". • `ReportOutOfDateAndUpToDate` : fournit en retour l'état des types "obsolète" et "actuel" :

Résultat

Lors de la détermination de version, les appareils du projet sont interrogés de haut en bas. La présence d'une instance d'une version de type issue de la bibliothèque indiquée dans les données de configuration de l'appareil est contrôlée pour chaque appareil. La méthode `UpdateCheck` fournit le résultat de la vérification de version selon un ordre hiérarchique.

Le tableau ci-après montre le résultat d'une vérification de version avec le paramètre `UpdateCheck.ReportOutOfDateAndUpToDate`:

Update check for: HMI_1				
	Update check for library element Screen_1 0.0.3			
		Out-of-date		
			\HMI_1\Screens	Screen_4 0.0.1
			\HMI_1\Screens	Screen_2 0.0.2
		Up-to-date		
			\HMI_1\Screens	Screen_1 0.0.3
			\HMI_1\Screens	Screen_10 0.0.3
Update check for: HMI_2				
	Update check of library element Screen_4 0.0.3			
		Out-of-date		
			\Screens folder1	Screen_02 0.0.1
			\Screens folder1	Screen_07 0.0.2
		Up-to-date		
			\Screens folder1	Screen_05 0.0.3
			\Screens folder1	Screen_08 0.0.3

Code du programme

Pour déterminer les versions de type d'une bibliothèque globale pour les instances dans le projet, modifiez le code de programme suivant :

```
public static void UpdateCheckOfGlobalLibrary(Project project, TiaPortal tiaPortal)
{
    GlobalLibrary globalLibrary = tiaPortal.GlobalLibraries.Open(@"SomePathHere");
    // check for out of date instances and report only out of date instances
    // in the returned feedback
    UpdateCheckResult result = globalLibrary.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateOnly);

    //Alternatively, check for out of date instances and report both out of date
    //and up to date instances in the returned feedback
    UpdateCheckResult alternateResult = globalLibrary.UpdateCheck(project,
UpdateCheckMode.ReportOutOfDateAndUpToDate);

    //Show result
    RecursivelyWriteMessages(result.Messages);
    // Alternatively, show result and access single message parts
    RecursivelyWriteMessageParts(result.Messages);
}
```

Pour déterminer les versions de type d'une bibliothèque de projet pour les instances dans le projet, modifiez le code de programme suivant :

```
public static void UpdateCheckOfProjectLibrary(Project project)
{
    // check for out of date instances and report only out of date instances
    // in the returned feedback
    UpdateCheckResult result =
project.ProjectLibrary.UpdateCheck(UpdateCheckMode.ReportOutOfDateOnly);

    //Alternatively, check for out of date instances and report both out of date
    //and up to date instances in the returned feedback
    UpdateCheckResult alternateResult =
project.ProjectLibrary.UpdateCheck(UpdateCheckMode.ReportOutOfDateAndUpToDate);

    //Show result
    RecursivelyWriteMessages(result.Messages);
    // Alternatively, show result und access single message parts
    RecursivelyWriteMessageParts(result.Messages);
}
```

Pour afficher le résultat de la détermination de version et parcourir les messages un par un, modifiez le code de programme suivant :

```
private void RecursivelyWriteMessages (UpdateCheckResultMessageComposition messages,
string indent = "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + message.Description);
        RecursivelyWriteMessages(message.Messages, indent);
    }
}
```

Pour accéder à certaines parties de message dans le résultat de la détermination de version, modifiez le code de programme suivant :

```
private void RecursivelyWriteMessageParts (UpdateCheckResultMessageComposition messages,
string indent= "")
{
    indent += "\t";
    foreach (UpdateCheckResultMessage message in messages)
    {
        Console.WriteLine(indent + "Full description: " + message.Description);
        foreach (KeyValuePair<string, string> messagePart in message.MessageParts)
        {
             // first level
             // part 1: device name
             // second level:
             // part 1: Name of the type in the global library
             // part 2: version of the type in the global library
             // third level:
             // part 1: title (either "Out-of-date" or "Up-to-date");
             // fourth level:
             // part 1: Path hierarchy to instance
             // part 2: Instance name in project
             // part 3: Version of the instance in the project
            Console.WriteLine(indent + "*Key: {0} Value:{1}", messagePart.Key,
messagePart.Value);
        }
     RecursivelyWriteMessageParts(message.Messages,indent);
    }
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.10.15.13 Actualiser un projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un dossier pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

L'interface Public API vous permet de mettre à jour des instances de types sélectionnés dans un dossier de type d'un projet.

Lors de l'actualisation, les instances utilisées dans le projet sont actualisées sur la base de la dernière version de type validée. Si vous actualisez les instances d'une bibliothèque globale, une synchronisation est d'abord exécutée.

Signature

Utilisez la méthode `UpdateProject` pour actualiser des instances.

Pour les classes implémentant l'interface `ILibraryType`, utilisez l'appel suivant :

```
void UpdateProject(IUpdateProjectScope updateProjectScope,
UpdatePathsMode updatePathsMode, DeleteUnusedVersionsMode
deleteUnusedVersionsMode)
```

Pour les classes implémentant l'interface `ILibrary`, utilisez l'appel suivant :

```
void UpdateProject(IEnumerable<ILibraryTypeOrFolderSelection>
selectedTypesOrFolders, IEnumerable <IUpdateProjectScope>
updateProjectScope, UpdatePathsMode updatePathsMode,
DeleteUnusedVersionsMode deleteUnusedVersionsMode)
```

Chaque appel est entré dans le fichier-journal du répertoire de projet.

Paramètre	Fonction
`IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders`	Indique les dossiers ou types devant être synchronisés ou dont les instances doivent être actualisées dans le projet.
`IUpdateProjectScope updateProjectScope` `IEnumerable <IUpdateProjectScope> updateProjectScope`	Indique dans le projet les objets dans lesquels les utilisations des instances sont actualisées. Les objets suivants sont pris en charge : • `PlcSoftware` • `HmiTarget`

Paramètre	Fonction
`UpdatePathsMode updatePathsMode`	Indique la réaction dans le cas où la structure des dossiers de la bibliothèque globale et celle de la bibliothèque du projet sont différentes. Le paramètre n'est évalué que si vous lancez l'actualisation du projet depuis une bibliothèque globale : • `UpdatePathsInTarget` : la structure des dossiers dans la bibliothèque cible est actualisée et le contenu déplacé en conséquence. • `KeepExistingPathsInTarget` : la structure des dossiers de la bibliothèque cible est conservée. Le contenu est actualisé. • `ThrowIfPathsConflict` : l'actualisation est annulée. Aucune modification n'a été effectuée. Une exception est déclenchée.
`DeleteUnusedVersionsMode deleteUnusedVersionsMode`	Indique la manipulation des versions de types qui ne sont plus utilisées : • `AutomaticallyDelete` : la version de type est supprimée. Seules les versions de types déterminées comme obsolètes lors de l'actualisation sont supprimées. • `DoNotDelete` : la version de type n'est pas supprimée.

Code du programme

Pour actualiser les instances de types sélectionnés au sein d'un dossier de type, modifiez le code de programme suivant :

```
private static void UpdateInstances(ILibrary myLibrary,
ILibraryTypeFoldersingleFolderContainingTypes, LibraryType singleType, PlcSoftware
plcSoftware, HmiTargethmiTarget)
{
    //Update Instances of multiple types (subset of types and folders)
    IUpdateProjectScope[] updateProjectScopes =
    {
        plcSoftware as IUpdateProjectScope, hmiTarget as IUpdateProjectScope
    };
    myLibrary.UpdateProject(new ILibraryTypeOrFolderSelection[] {singleType,
singleFolderContainingTypes}, updateProjectScopes,
UpdatePathsMode.KeepExistingPathsInTarget, DeleteUnusedVersionsMode.AutomaticallyDelete);
    //Update Instances of multiple types (all types in library)
    myLibrary.UpdateProject(new[] {myLibrary.TypeFolder}, updateProjectScopes,
UpdatePathsMode.UpdatePathsInTarget, DeleteUnusedVersionsMode.AutomaticallyDelete);
}
```

7.10.15.14 Actualiser une bibliothèque

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un dossier pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

L'interface Public API prend en charge les actualisations suivantes dans la bibliothèque de projet :

- Synchronisation de tous les types entre la bibliothèque globale et la bibliothèque de projet
- Synchronisation des types sélectionnés entre la bibliothèque globale et la bibliothèque de projet

Seul le sens de synchronisation d'une bibliothèque globale vers la bibliothèque de projet est pris en charge lors de la synchronisation de types. La structure des dossiers peut être adaptée en option lors de la synchronisation. Les types à actualiser sont déterminés et actualisés à l'aide de leur GUID :

- Si un type d'une bibliothèque comporte une version de type manquant dans la bibliothèque devant être actualisée, la version de type est copiée.
- L'opération est annulée et une Exception est émise, si un type d'une bibliothèque comporte une version de type avec les propriétés suivantes :
  - La version de type est présente dans la bibliothèque devant être actualisée.
  - La version de type a le même numéro de version
  - La version de type a un GUID différent.

Signature

Pour synchroniser les versions de types, utilisez la méthode `UpdateLibrary`.

Pour les classes implémentant l'interface `LibraryType`, utilisez l'appel suivant :

```
void UpdateLibrary(ILibrary targetLibrary, UpdatePathsMode
updatePathsMode, DeleteUnusedVersionsMode deleteUnusedVersionsMode)
```

Pour les classes implémentant l'interface `ILibrary`, utilisez l'appel suivant :

```
void UpdateLibrary(IEnumerable<LibraryTypeOrFolderSelection>
selectedTypesOrFolders, ILibrary targetLibrary, UpdatePathsMode
updatePathsMode, DeleteUnusedVersionsMode deleteUnusedVersionsMode)
```

Paramètre	Fonction
`IEnumerable<ILibraryTypeOrFolderSelection> selectedTypesOrFolders`	Indique les dossiers ou types devant être synchronisés ou dont les instances doivent être actualisées dans le projet.
`ILibrary targetLibrary`	Indique la bibliothèque dont le contenu doit être synchronisé avec une bibliothèque globale. Si la bibliothèque source et la bibliothèque cible sont identiques, une exception est déclenchée. Si la bibliothèque cible est une bibliothèque globale, une exception est déclenchée.
`UpdatePathsMode updatePathsMode`	Indique la réaction dans le cas où la structure des dossiers de la bibliothèque globale et celle de la bibliothèque du projet sont différentes. Le paramètre n'est évalué que si vous lancez l'actualisation du projet depuis une bibliothèque globale : • `UpdatePathsInTarget` : la structure des dossiers dans la bibliothèque cible est actualisée et le contenu déplacé en conséquence. • `KeepExistingPathsInTarget` : la structure des dossiers de la bibliothèque cible est conservée. Le contenu est actualisé. • `ThrowIfPathsConflict` : l'actualisation est annulée. Aucune modification n'a été effectuée. Une exception est déclenchée.
`DeleteUnusedVersionsMode deleteUnusedVersionsMode`	Indique la manipulation des versions de types qui ne sont plus utilisées : • `AutomaticallyDelete` : la version de type est supprimée. Seules les versions de types déterminées comme obsolètes lors de l'actualisation sont supprimées. • `DoNotDelete` : la version de type n'est pas supprimée.

Code du programme

Pour synchroniser un type d'une bibliothèque de projet avec une bibliothèque globale, modifiez le code de programme suivant :

```
sourceType.UpdateLibrary(projectLibrary, UpdatePathsMode.KeepExistingPathsInTarget,
DeleteUnusedVersionsMode.DoNotDelete);
```

Pour synchroniser des types sélectionnés dans un dossier de type entre une bibliothèque globale et la bibliothèque de projet, modifiez le code de programme suivant :

```
globalLibrary.UpdateLibrary(new[]{globalLibrary.TypeFolder}, projectLibrary,
UpdatePathsMode.KeepExistingPathsInTarget, DeleteUnusedVersionsMode.DoNotDelete);
```

7.10.15.15 Supprimer les contenus de bibliothèque

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)
- Vous avez accès à la bibliothèque requise.
  Voir Accéder aux bibliothèques (Page 113).
- Vous avez accès à un dossier pour les types.
  Voir Accéder aux dossiers dans une bibliothèque (Page 115).

Utilisation

L'interface Public API vous permet de supprimer les contenus suivants dans la bibliothèque du projet :

- Types
- Version de type
- Dossiers personnalisés pour types
- Copies maîtresse
- Dossiers personnalisés pour copies maîtresse

Remarque

Suppression de types et de dossiers avec des types personnalisés

Si vous souhaitez supprimer un type ou un dossier avec des types personnalisés, les "Règles de suppression de versions" doivent être respectées. Vous pouvez supprimer un dossier de type vide à tout moment.

Remarque

Règles de suppression de versions

Seules les versions au statut "Committed" peuvent être supprimées. Pour la suppression de versions, les règles à respecter sont les suivantes :

- Si vous venez de créer une version au statut "`InWork`" depuis une version au statut "`Committed`", vous ne pouvez supprimer la version au statut "`Committed`" que lorsque la nouvelle version est annulée ou obtient le statut "`Committed`".
- Si un type ne possède qu'une seule version, le type est également supprimé.
- Si la version A dépend de la version B d'un autre type, supprimez d'abord la version A, puis la version B.
- Si des instances de la version A existent, vous ne pouvez supprimer la version A que si vous supprimez également les instances. De plus, si une instance se trouve dans une copie maîtresse, celle-ci est également supprimée.

Code du programme

Pour supprimer des types ou des dossiers avec des types personnalisés, modifiez le code de programme suivant :

```
public static void DeleteMultipleTypesOrTypeUserFolders(ILibrary library)
{
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    LibraryType t2 = library.TypeFolder.Types.Find("type2");
    LibraryTypeUserFolder f1 = library.TypeFolder.Folders.Find("folder1");
    t1.Delete();
    t2.Delete();
    f1.Delete();
}
```

Pour supprimer un type ou dossier avec des types personnalisés en particulier, modifiez le code de programme suivant :

```
public static void DeleteSingleTypeOrTypeUserFolder(ILibrary library)
{
    //Delete a single type
    LibraryType t1 = library.TypeFolder.Types.Find("type1");
    t1.Delete();

    //Delete a single folder
    ILibraryTypeFolder parentFolder = library.TypeFolder;
    LibraryTypeUserFolder f1 = parentFolder.Folders.Find("folder1");
    f1.Delete();
}
```

Pour supprimer une version, modifiez le code de programme suivant :

```
public static void DeleteVersion(ILibrary library)
{
    LibraryType singleType = library.TypeFolder.Types.Find("type1");
    LibraryTypeVersion version1 = singleType.Versions.Find(new System.Version(1, 0, 0));
    version1.Delete();
}
```

Pour supprimer une copie maîtresse ou un dossier avec des copies maîtresses personnalisées, modifiez le code de programme suivant :

```
public static void DeleteMasterCopies(ILibrary library)
{
    // Delete master copy
    MasterCopy masterCopy = library.MasterCopyFolder.MasterCopies.Find("myMasterCopy");
    masterCopy.Delete();

    // Delete master copy user folder
    MasterCopyUserFolder masterUserFolder =
library.MasterCopyFolder.Folders.Find("myFolder");
    masterUserFolder.Delete();
}
```

Voir aussi

Accéder à des modèles de copie (Page 125)

7.10.16 Lire des attributs liés au projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Cette fonction vous permet d'appeler des attributs liés au projet issus de la Public API. Les informations fournies contiennent les propriétés du projet, l'historique du projet et les produits utilisés par le projet.

Propriétés du projet

Les propriétés du projet fournissent les informations suivantes :

Nom d'attribut	Type de données	Accessible en écriture	Description
Author	System.String	r/o	Auteur du projet.
Name	System.String	r/o	Nom du projet.
Path	System.String	r/o	Chemin absolu du projet.
CreationTime	System.DateTime	r/o	Date et heure auxquelles le projet a été créé.
LastModified	System.DateTime	r/o	Date et heure auxquelles le projet a été modifié pour la dernière fois.
LastModifiedBy	System.String	r/o	Auteur de la dernière modification.
Version	System.String	r/o	Version du projet.
Comment	Siemens.Engineering.MultilingualText	r/o	Commentaire du projet.
Copyright	System.String	r/o	Mention de copyright du projet.
Family	System.String	r/o	Famille du projet.
Size	System.Int64	r/o	Taille du projet, en Ko.
Languages	Siemens.Engineering.LanguageComposition	r/o	Langues utilisées dans le projet.

Pour accéder aux attributs liés au projet, modifiez le code de programme suivant :

```
Project project = ...;
string author = project.Author;
string name = project.Name;
string path = project.Path;
DateTime creationTime = project.CreationTime;
DateTime modificationTime = project.LastModified;
string lastModifiedBy = project.LastModifiedBy;
string version = project.Version;
MultilingualText comment = project.Comment;
string copyright = project.Copyright;
string family = project.Family;
Int64 size = project.Size;
LanguageComposition languages = project.Languages;
```

Pour énumérer les langues du projet, modifiez le code de programme suivant :

```
Project project = ...;
LanguageComposition languages = project.Languages;
foreach (Language language in languages)
{
    CultureInfo lang = language.Culture;
}
```

Pour appeler le texte de commentaire avec la culture actuelle, modifiez le code de programme suivant :

```
Project project = ...;
MultilingualText projectComment = project.Comment;
string comment = projectComment.GetText(CultureInfo.CurrentCulture);
```

Historique du projet

L'historique du projet regroupe des objets du type `HistoryEntry`, qui contiennent les informations suivantes :

Nom d'attribut	Type de données	Accessible en écriture	Description
Text	System.String	r/o	Description de l'événement.
DateTime	System.DateTime	r/o	Date et heure auxquelles l'événement s'est produit.

Pour énumérer HistoryEntries et accéder à ses propriétés, modifiez le code de programme suivant :

```
Project project = ...;
HistoryEntryComposition historyEntryComposition = project.HistoryEntries;
foreach (HistoryEntry historyEntry in historyEntryComposition)
{
    string entryText = historyEntry.Text;
    DateTime entryTime = historyEntry.DateTime;
}
```

Remarque

La propriété Texte de `HistoryEntry` contient une chaîne de caractères dans la même langue que celle de l'interface utilisateur. Lorsqu'une application Openness est attachée à un TIA Portal sans interface utilisateur, la chaîne de caractères est fournie en anglais par défaut.

Produits utilisés

L'objet `UsedProduct` contient les informations suivantes :

Nom d'attribut	Type de données	Accessible en écriture	Description
Name	System.String	r/o	Nom du projet utilisé.
Version	System.String	r/o	Version du projet.

Pour énumérer `UsedProduct` et accéder à ses propriétés, modifiez le code de programme suivant :

```
Project project = ...;
UsedProductComposition usedProductComposition = project.UsedProducts;
foreach (UsedProduct usedProduct in usedProductComposition)
{
    string productName = usedProduct.Name; string productVersion = usedProduct.Version;
}
```

7.10.17 Suppression d'un graphique du projet

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour supprimer une bibliothèque de graphiques, modifiez le code de programme suivant :

```
//Deletes a single project graphic entry
public static void DeletesSingleProjectGraphicEntry(Project project)
{
    MultiLingualGraphicComposition graphicsAggregation = project.Graphics;
    MultiLingualGraphic graphic = graphicsAggregation.Find("Graphic XYZ");
    graphic.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.18 Enregistrer le projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Utilisez la méthode `project.Save()` pour enregistrer un projet.

Code du programme

Pour ouvrir et enregistrer un projet, modifiez le code de programme suivant :

```
public static void SaveProject(Project project)
{
    //Use the code in the try block to open and save a project
    try
    {
        project = tiaPortal.Projects.Open(@"Some\Path\MyProject.ap14");
        //begin of code for further implementation
        //...
        //end of code
        project.Save();
    }
        //Use the code in the final block to close a project
    finally
    {
        project.Close();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.19 Déterminer le statut d'un API

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez déterminer l'état d'un API ou de tous les API au sein d'un projet.

Openness distingue les états suivants :

- Offline
- l'API est connecté ("la connexion est établie")
- En ligne
- l'API n'est pas connecté ("la connexion est coupée")
- Incompatible
- accès impossible
- Protégé

Code du programme

Pour déterminer l'état d'un API, modifiez le code de programme suivant :

```
public static OnlineState GetOnlineState(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    return onlineProvider.State;
}
```

Pour déterminer l'état de tous les API dans un projet, modifiez le code de programme suivant :

```
public static void DetermineOnlineStateOfAllProjectDevices(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                OnlineState state = onlineProvider.State;
            }
        }
    }
}
```

7.10.20 Comparer le logiciel de l'API

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)

Utilisation

Pour déterminer la divergence entre les logiciels de deux appareils, vous disposez des possibilités suivantes :

- Comparaison entre les logiciels de deux API configurés
- Comparaison entre le logiciel d'un API et la bibliothèque de projet
- Comparaison entre le logiciel d'un API et la bibliothèque globale
- Comparaison entre le logiciel d'un API et la copie principale d'un API
- Comparaison entre le logiciel d'un API configuré et le logiciel d'un API connecté à l'état "En ligne"

Signature

Utilisez la méthode `CompareTo` ou `CompareToOnline` pour la comparaison.

```
public CompareResult CompareTo (ITargetComparable compareTarget)
```

```
public CompareResult CompareToOnline ()
```

Valeur de retour/paramètre	Fonction
`CompareResult compareResult`	Fournit en retour le résultat de comparaison : • `FolderContentsDifferent` : le contenu des dossiers comparés diffère. • `FolderContentsIdentical` : le contenu des dossiers comparés est identique. • `ObjectsDifferent` : le contenu des objets comparés diffère. • `ObjectsIdentical` : le contenu des objets comparés est identique. • `LeftMissing` : l'objet n'est pas compris dans l'objet à partir duquel la comparaison a été lancée. • `RightMissing` : l'objet n'est pas compris dans l'objet auquel la comparaison s'applique.
`ITargetComparable compareTarget`	Liste d'objets comparables.

Code du programme

Pour afficher le résultat de la comparaison, modifiez le code de programme suivant :

```
private static void WriteResult(CompareResultElement compareResultElement, string indent)
{
    Console.WriteLine("{0}<{1}> <{2}> <{3}> <{4}> <{5}> ",
    indent,
    compareResultElement.LeftName,
    compareResultElement.ComparisonResult,
    compareResultElement.RightName,
    compareResultElement.PathInformation,
    compareResultElement.DetailedInformation);
    WriteResult(compareResultElement.Elements, indent);
}
private static void WriteResult (IEnumerable<CompareResultElement> compareResultElements,
string indent)
{
    indent += " ";
    foreach (CompareResultElement compareResultElement in compareResultElements)
    {
        WriteResult(compareResultElement, indent);
    }
}
```

Pour comparer les logiciels des appareils, modifiez le code de programme suivant :

```
private static void CompareTwoOfflinePlcs(PlcSoftware plcSoftware0, PlcSoftware
plcSoftware1)
{
    if (plcSoftware0 != null && plcSoftware1 != null)
    {
        CompareResult compareResult = plcSoftware0.CompareTo(plcSoftware1);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Pour comparer le logiciel d'un API avec la bibliothèque de projet, modifiez le code de programme suivant :

```
private static void ComparePlcToProjectLibrary(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(project.ProjectLibrary);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Pour comparer le logiciel d'un API avec la bibliothèque globale, modifiez le code de programme suivant :

```
private static void ComparePlcToGlobalLibrary(PlcSoftware plcSoftware, GlobalLibrary
globalLibrary)
{
    if (plcSoftware != null && globalLibrary != null)
    {
        CompareResult compareResult = plcSoftware.CompareTo(globalLibrary);
        WriteResult(compareResult.RootElement, String.Empty);
    }
}
```

Pour comparer le logiciel d'un API avec une copie maîtresse, modifiez le code de programme suivant :

```
private static void ComparePlcToMasterCopy(Project project, PlcSoftware plcSoftware)
{
    if (project != null && plcSoftware != null)
    {
        CompareResult compareResult =
plcSoftware.CompareTo(project.ProjectLibrary.MasterCopyFolder.MasterCopies[0]);
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

Pour comparer le logiciel d'un API avec le logiciel d'un API connecté, modifiez le code de programme suivant :

```
private static void ComparePlcToOnlinePlc(PlcSoftware plcSoftware)
{
    if (plcSoftware != null)
    {
        CompareResult compareResult = plcSoftware.CompareToOnline();
        WriteResult(compareResult.RootElement, string.Empty);
    }
}
```

7.10.21 Accéder aux paramètres d'une liaison en ligne

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface Public API vous permet de définir les paramètres pour une liaison en ligne :

- Enumérer les types de connexion disponibles avec un API
- Enumérer les interfaces disponibles avec un API
- Enumérer les emplacements affectés
- Enumérer les adresses disponibles des sous-réseaux et passerelles
- Définir les paramètres de liaison

Code de programme : déterminer les paramètres de liaison

Pour énumérer les modes de liaison, interfaces de PC et emplacements disponibles, modifiez le code de programme suivant :

```
public static void EnumerateConnectionModesOfPLC(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null)
    {
       return; // Only cpu device items can provide OnlineProvider service
    }
    // Accessing connection configuration object
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    // Now access connection configuration members
    foreach (ConfigurationMode mode in configuration.Modes)
    {
        Console.WriteLine("Mode name:{0}", mode.Name);
        foreach (ConfigurationPcInterface pcInterface in mode.PcInterfaces)
        {
            Console.WriteLine("PcInterface name:{0}", pcInterface.Name);
            Console.WriteLine("PcInterface number:{0}", pcInterface.Number);
            foreach (ConfigurationTargetInterface targetInterface in
pcInterface.TargetInterfaces)
            {
                Console.WriteLine("TargetInterface:{0}", targetInterface.Name);
            }
        }
    }
}
```

Vous pouvez aussi accéder à un type de connexion et une interface de PC par le nom :

```
public static ConfigurationTargetInterface
GetTargetInterfaceForOnlineConnection(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find("PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    return slot;
}
```

Pour énumérer les adresses disponibles des sous-réseaux et passerelles sur une interface de PC, modifiez le code de programme suivant :

```
public static void EnumeratingPCInterfaceSubnetsAndGateways(ConfigurationPcInterface
pcInterface)
{
    foreach (ConfigurationSubnet subnet in pcInterface.Subnets)
    {
        Console.WriteLine("Subnet name:{0}", subnet.Name);
        foreach (ConfigurationGateway gateway in subnet.Gateways)
        {
            //Get the name of the gateway:
            Console.WriteLine("Gateway name:{0}", gateway.Name);
            //Get the IP address of each gateway:
            foreach (ConfigurationAddress gatewayAddress in gateway.Addresses)
            {
                Console.WriteLine("Gateway Address:{0}", gatewayAddress.Name);
            }
        }
    }
}
```

Vous pouvez également accéder aux sous-réseaux et passerelles par le nom ou l'adresse IP :

```
public static void AccessSubnetAndGatewayOfPCInterface(ConfigurationPcInterface
pcInterface)
{
    ConfigurationSubnet subnet = pcInterface.Subnets.Find("PN/IE_1");
    ConfigurationAddress subnetAddress = subnet.Addresses.Find("192.168.0.1");
    ConfigurationGateway gateway = subnet.Gateways.Find("Gateway 1");
    ConfigurationAddress gatewayAddress = gateway.Addresses.Find("192.168.0.2");
}
```

Code de programme : Définir les paramètres de liaison

Remarque

La définition des paramètres de liaison écrase tous les paramètres de liaison définis auparavant. Si vous avez déjà défini les paramètres de liaison directement dans le portail TIA, vous n'avez pas besoin d'appeler `ApplyConfiguration`. Si une liaison en ligne existe déjà vers un API pendant l'appel de `ApplyConfiguration`, une exception sera déclenchée.

Pour définir les paramètres d'emplacement, modifiez le code de programme suivant :

```
public static void SetConnectionWithSlot(OnlineProvider onlineProvider)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationTargetInterface slot = pcInterface.TargetInterfaces.Find("2 X3");
    configuration.ApplyConfiguration(slot);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Pour définir les paramètres d'adresse de passerelles, modifiez le code de programme suivant :

```
public static void SetConnectionWithGatewayAddress(OnlineProvider onlineProvider, string
subnetName, string gatewayAddressName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddress gatewayAddress = subnet.Addresses.Find(gatewayAddressName);
    configuration.ApplyConfiguration(gatewayAddress);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

Pour définir les paramètres d'adresse de sous-réseaux, modifiez le code de programme suivant :

```
public static void SetConnectionWithSubnetAddress(OnlineProvider onlineProvider, string
subnetName)
{
    ConnectionConfiguration configuration = onlineProvider.Configuration;
    ConfigurationMode mode = configuration.Modes.Find(@"PN/IE");
    ConfigurationPcInterface pcInterface = mode.PcInterfaces.Find("PLCSIM", 1);
    // or network pc interface that is connected to plc
    ConfigurationSubnet subnet = pcInterface.Subnets.Find(subnetName);
    ConfigurationAddressComposition addresses = subnet.Addresses;
    configuration.ApplyConfiguration(addresses[0]);
    // After applying configuration, you can go online
    onlineProvider.GoOnline();
}
```

7.10.22 Etablir ou interrompre une liaison en ligne à l'API

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Tous les appareils sont énumérés.
  Voir Enumérer et appeler des éléments d'appareils (Page 95).

Utilisation

Vous pouvez établir la liaison en ligne à un API ou interrompre une liaison en ligne existante.

Code du programme

Pour établir ou interrompre la liaison en ligne à un API, modifiez le code de programme suivant :

```
public static void SetOnlineConnection(DeviceItem deviceItem)
{
    OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
    if (onlineProvider == null) { return; }
    // Go online
    if (onlineProvider.Configuration.IsConfigured)
    {
        onlineProvider.GoOnline();
    }
   // Go offline
   onlineProvider.GoOffline();
}
```

Vous pouvez également établir ou interrompre les liaisons en ligne à tous les API disponibles dans un projet.

```
public static void SetOnlineConnectionForAllPLCs(Project project)
{
    foreach (Device device in project.Devices)
    {
        foreach (DeviceItem deviceItem in device.DeviceItems)
        {
            OnlineProvider onlineProvider = deviceItem.GetService<OnlineProvider>();
            if (onlineProvider != null)
            {
                // Establish online connection to PLC:
                onlineProvider.GoOnline();
                // Disconnect online connection to PLC:
                onlineProvider.GoOffline();
            }
        }
    }
}
```

7.10.23 Fermer un projet

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour fermer un projet, modifiez le code de programme suivant :

```
public static void CloseProject(Project project)
{
    project.Close();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.10.24 Prise en charge de l'autodescription pour attributs, navigateurs, actions et services

Utilisation

Dans Openness, chaque IEngineeringServiceProvider de la Public API décrit ses capacités pour de potentiels appels.

Prise en charge de l'autodescription pour IEngineeringObject

Nom de la méthode	Valeurs retournées
GetCompositionInfos	Fournit en retour une bibliothèque d'objets du type EngineeringCompositionInfo, qui décrivent les différentes compositions de ces objets. La section qui suit propose une description de EngineeringCompositionInfo.
GetAttributeInfos	Fournit en retour une bibliothèque d'objets du type EngineeringAttributeInfo, qui décrivent les différents attributs de ces objets. La section qui suit propose une description deEngineeringAttributeInfo.
GetInvocationInfos	Fournit en retour une bibliothèque d'objets du type EngineeringInvocationInfo, qui décrivent les différentes actions de ces objets. La section qui suit propose une description de EngineeringInvocationInfo.

Prise en charge de l'autodescription pour IEngineeringServiceProvider

Nom de la méthode	Valeurs retournées
GetServiceInfos	Fournit en retour une bibliothèque d'objets du type EngineeringServiceInfo, qui décrivent les différents services de ces objets. La section qui suit propose une description de EngineeringServiceInfo.

Classe EngineeringCompositionInfo

Nom de la méthode	Valeurs retournées
Name	Nom de la composition.

Classe EngineeringAttributeInfo

Nom de la méthode	Valeurs retournées
AccessMode	Niveau d'accès pris en charge par l'attribut. La section qui suit propose une description détaillée de cet attribut.
Name	Nom de l'attribut.

Classe EngineeringInvocationInfo

Nom de la méthode	Valeurs retournées
Name	Nom de l'action.
ParameterInfos	Une bibliothèque d'objets du type EngineeringInvocationParameterInfo, qui décrivent les paramètres requis éventuellement pour l'action. La section qui suit propose une description de EngineeringInvocationParameterInfo.

Classe EngineeringServiceInfo

Nom de la méthode	Valeurs retournées
Type	Type de service comme objet System.Type.

Enum AccessMode

Valeur d'énumération	Valeurs retournées
None	Option invalide.
Read	L'attribut peut être lu.
Write	L'attribut peut être écrit.

Classe EngineeringInvocationParameterInfo

Nom de la méthode	Valeurs retournées
Name	Nom du paramètre.
Type	Type du paramètre comme objet System.Type.

Code de programme

AccessMode est une énumération de mémentos dont les valeurs peuvent être combinées comme dans le code de programme suivant :

```
EngineeringAttributeAccessMode value = EngineeringAttributeAccessMode.Read|
EngineeringAttributeAccessMode.Write;
```

Pour rechercher tous les attributs d'un IEngineeringObject et apporter des modifications au mode d'accès à ces derniers, modifiez le code de programme suivant :

```
...
IEngineeringObject engineeringObject = ...;
IList<EngineeringAttributeInfo> attributeInfos = engineeringObject.GetAttributeInfos();
foreach(EngineeringAttributeInfo attributeInfo in attributeInfos)
{
    switch (attributeInfo.AccessMode)
        {
        case EngineeringAttributeAccessMode.Read:
            ...
            break;
        case EngineeringAttributeAccessMode.Write:
            ...
            break;
        case EngineeringAttributeAccessMode.Read|EngineeringAttributeAccessMode.Write:
            ...
            break;
        }
}
...
```

7.11 Fonctions sur les données d'un appareil HMI

7.11.1 Vues

7.11.1.1 Créer des dossiers de vues personnalisés

Condition

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour créer un dossier de vues personnalisé, modifiez le code de programme suivant :

```
private static void CreateScreenFolder(HmiTarget hmitarget)
//Creates a screen folder
{
    ScreenUserFolder myCreatedFolder =
hmitarget.ScreenFolder.Folders.Create("myScreenFolder");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.1.2 Supprimer la vue d'un dossier

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Remarque

Vous ne pouvez pas supprimer une fenêtre permanente. Une fenêtre permanente est une vue système toujours existante.

Code du programme

Pour supprimer une vue d'un certain dossier, modifiez le code de programme suivant :

```
public static void DeleteScreenFromFolder(HmiTarget hmiTarget)
{
    ScreenUserFolder screenUserFolder =
hmiTarget.ScreenFolder.Folders.Find("myScreenFolder");
    ScreenComposition screens = screenUserFolder.Screens;
    Screen screen = screens.Find("myScreenName");
    if (screen != null)
    {
        screen.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.1.3 Supprimer un modèle de vue d'un dossier

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer un modèle de vue d'un certain dossier, modifiez le code de programme suivant :

```
private static void DeleteScreenTemplateFromFolder(HmiTarget hmiTarget)
{
    string templateName = "MyScreenTemplate";
    ScreenTemplateUserFolder folder =
hmiTarget.ScreenTemplateFolder.Folders.Find("myScreenTemplateFolder");
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if (template != null)
    {
        template.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.1.4 Supprimer toutes les vues d'un dossier

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Remarque

Vous ne pouvez pas supprimer une fenêtre permanente. Une fenêtre permanente est une vue système toujours existante.

Code du programme

Pour supprimer toutes les vues d'un certain dossier, modifiez le code de programme suivant :

```
private static void DeleteAllScreensFromFolder(HmiTarget hmitarget)
//Deletes all screens from a user folder or a system folder
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("myScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    List<Screen> list = new List<Screen>();
    foreach(Screen screen in screens)
    {
        list.Add(screen);
    }
    foreach (Screen screen in list)
    {
        screen.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.2 Cycles

7.11.2.1 Suppression de cycle

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Utilisation

Vous ne pouvez pas supprimer les cycles prédéfinis.

À l'aide de la composition dans le modèle objet (composition count) du cycle concerné, vous pouvez déterminer si des cycles ont effectivement été supprimés. Il n'est plus possible d'accéder à ces cycles.

Code du programme

Pour supprimer un cycle d'un appareil IHM, modifiez le code de programme suivant :

```
public static void DeleteCycle(HmiTarget hmiTarget)
{
    CycleComposition cycles = hmiTarget.Cycles;
    Cycle cycle = cycles.Find("myCycle");
    cycle.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.3 Listes de textes

7.11.3.1 Suppression de la liste de textes

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer une liste de textes sélectionnée et toutes les entrées de liste correspondantes d'un appareil IHM, modifiez le code de programme suivant :

```
public static void DeleteTextList(HmiTarget hmiTarget)
{
    TextListComposition textLists = hmiTarget.TextLists;
    TextList textList = textLists.Find("myTextList");
    textList.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.4 Listes de graphiques

7.11.4.1 Suppression d'une liste de graphiques

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer une liste de graphiques sélectionnée et toutes les entrées de liste correspondantes d'un appareil IHM, modifiez le code de programme suivant :

```
private static void DeleteGraphicList(HmiTarget hmiTarget)
{
    GraphicListComposition graphicLists = hmiTarget.GraphicLists;
    GraphicList graphicList = graphicLists.Find("myGraphicList");
    graphicList.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.5 Connexions

7.11.5.1 Suppression de la liaison

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer une liaison de communication sélectionnée d'un appareil IHM, modifiez le code de programme suivant :

```
private static void DeleteConnection(HmiTarget hmiTarget)
{
    ConnectionComposition connections = hmiTarget.Connections;
    Connection connection = connections.Find("HMI_connection_1");
    connection.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.6 Table des variables

7.11.6.1 Générer des dossiers personnalisés pour variables IHM

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour créer un dossier personnalisé pour variables HMI, modifiez le code de programme suivant :

```
private static void CreateUserfolderForHMITags(HmiTarget hmitarget)
// Creates an HMI tag user folder
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagUserFolder myCreatedFolder = folder.Folders.Create("MySubFolder");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.6.2 Enumérer les variables d'une table de variables IHM

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour énumérer toutes les variables d'une table de variables IHM, modifiez le code de programme suivant :

```
private static void EnumerateTagsInTagtable(HmiTarget hmitarget)
// //Enumerates all tags of a tag table
{
    TagTable table = hmitarget.TagFolder.TagTables.Find("MyTagtable");
    // Alternatively, you can access the default tag table:
    // TagTable defaulttable = hmitarget.TagFolder.DefaultTagTable;

    TagComposition tagComposition = table.Tags;
    foreach (Tag tag in tagComposition)
    {
        // Add your code here
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.6.3 Suppression de variables individuelles d'une table de variables IHM

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour supprimer une variable déterminée d'une table des variables IHM, modifiez le code de programme suivant :

```
private static void DeleteATag(HmiTarget hmiTarget)
{
    string tagName = "MyTag";
    TagTable defaultTagTable = hmiTarget.TagFolder.DefaultTagTable;
    TagComposition Variablen = defaultTagTable.Tags;
    Tag tag = tags.Find(tagName);
    tag.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.6.4 Supprimer une table de variables d'un dossier

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer un tableau des variables d'un certain dossier, modifiez le code de programme suivant :

```
private static void DeleteTagTable(HmiTarget hmiTarget)
{
    string tableName = "myTagTable";
    TagSystemFolder tagSystemFolder = hmiTarget.TagFolder;
    TagTableComposition tagTables = tagSystemFolder.TagTables;
    TagTable tagTable = tagTables.Find(tableName);
    tagTable.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.7 Scripts VB

7.11.7.1 Créer des dossiers personnalisés pour les scripts

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour créer un sous-dossier personnalisé pour scripts dans un dossier système ou un autre dossier personnalisé, modifiez le code de programme suivant :

```
private static void CreateFolderInScriptfolder(HmiTarget hmitarget)
//Creates a script user subfolderVBScriptSystemFolder
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbScriptFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbScriptSubFolder = vbScriptFolder.Folders.Create("mySubfolder");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.7.2 Supprimer les scripts VB d'un dossier

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un appareil IHM existe dans le projet.

Code du programme

Pour supprimer un script VB d'un certain dossier, modifiez le code de programme suivant :

```
private static void DeleteVBScriptFromScriptFolder(HmiTarget hmitarget)
//Deletes a vbscript from a script folderVBScriptSystemFolder
{
    VBScriptUserFolder vbscriptfolder =
hmitarget.VBScriptFolder.Folders.Find("MyScriptFolder");
    var vbScripts = vbscriptfolder.VBScripts;
    if (null != vbScripts)
    {
        var vbScript = vbScripts.Find("MyScript");
        vbScript.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.11.8 Supprimer le dossier personnalisé d'un pupitre opérateur

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code de programme

Pour supprimer un dossier personnalisé d'un pupitre opérateur, modifiez le code de programme suivant :

```
HmiTarget hmiTarget = ...;
ScreenUserFolder screenUserGroup = hmiTarget.ScreenFolder.Folders.Find("MyUserFolder");
screenUserGroup.Delete();
```

7.12 Fonctions sur les données d'un appareil API

7.12.1 Blocs

7.12.1.1 Interroger le groupe "Blocs de programme"

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un API est décelé dans le projet.

Code du programme

Pour interroger le groupe "Blocs de programme", modifiez le code de programme suivant :

```
private static void GetBlockGroupOfPLC(PlcSoftware plcsoftware)
//Retrieves the system group of a block
{
    PlcBlockSystemGroup blockGroup = plcsoftware.BlockGroup;
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.2 Enumérer les groupes Blocs personnalisés

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un API est décelé dans le projet.

Utilisation

Les sous-groupes compris sont considérés comme récurrents lors de l'énumération.

Code du programme : Enumérer tous les groupes

Pour énumérer les groupes Blocs personnalisés, modifiez le code de programme suivant :

```
//Enumerates all block user groups including sub groups
private static void EnumerateAllBlockGroupsAndSubgroups(PlcSoftware plcsoftware)
{
    foreach (PlcBlockUserGroup blockUserGroup in plcsoftware.BlockGroup.Groups)
    {
        EnumerateBlockUserGroups(blockUserGroup);
    }
}

private static void EnumerateBlockUserGroups(PlcBlockUserGroup blockUserGroup)
{
    foreach (PlcBlockUserGroup subBlockUserGroup in blockUserGroup.Groups)
    {
        EnumerateBlockUserGroups(subBlockUserGroup);
        // recursion
    }
}
```

Code du programme : Accéder à un groupe

Pour accéder à un groupe de blocs personnalisé sélectionné, modifiez le code de programme suivant :

```
//Gives individual access to a specific block user group
private static void AccessBlockusergroup(PlcSoftware plcsoftware)
{
    PlcBlockUserGroupComposition plcBlockUserGroupComposition
=plcsoftware.BlockGroup.Groups;
    PlcBlockUserGroup plcBlockUserGroup =
plcBlockUserGroupComposition.Find("MyUserfolder");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.3 Enumérer tous les blocs

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Un API est décelé dans le projet.

Utilisation

Il est possible d'accéder de manière ciblée à un bloc de programme si son nom est connu.

Code du programme : Enumérer tous les blocs

Pour énumérer les blocs de tous les groupes Blocs, modifiez le code de programme suivant :

```
private static void EnumerateAllBlocks(PlcSoftware plcsoftware)
//Enumerates all blocks
{
    foreach (PlcBlock block in plcsoftware.BlockGroup.Blocks)
    {
        // Do something...
    }
}
```

Code du programme : Accéder à un bloc déterminé

Pour accéder à un bloc donné, modifiez le code de programme suivant :

```
private static void AccessASingleBlock(PlcSoftware plcsoftware)
//Gives individual access to a block
{
    // The parameter specifies the name of the block
    PlcBlock block = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.4 Interroger les informations d'un bloc/type de données utilisateur

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Public API prend en charge l'interrogation des informations suivantes pour le programme et les blocs de données et pour les types de données utilisateur :

- Horodatage au format UTC
  L'horodatage fournit les informations suivantes :
  - Quand le bloc a-t-il été compilé pour la dernière fois ?
  - Quand le bloc a-t-il été modifié pour la dernière fois ?
- Attribut "Consistency"
  L'attribut « Consistency" est mis sur "True" dans les cas suivants :
  - Le bloc a été correctement compilé.
  - Le bloc n'a pas été modifié depuis la compilation.
  - Aucune modification ayant impliqué une nouvelle compilation n'a été apportée aux objets externes.
- Langage de programmation utilisé (uniquement programme et de blocs de données)
- Numéro de bloc
- Nom de bloc
- Auteur du bloc
- Famille de bloc
- Titre du bloc
- Version du bloc

Pour plus d'informations, voir Blocs et types de modèle d'objet Openness (Page 50).

Code du programme

Pour interroger les informations susmentionnées, modifiez le code de programme suivant :

```
private static void GetPlcBlockInformation(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    //Read information DateTime compileDate = plcBlock.CompileDate;
    DateTime modifiedDate = plcBlock.ModifiedDate;
    bool isConsistent = plcBlock.IsConsistent;
    int blockNumber = plcBlock.Number;
    string blockName = plcBlock.Name;
    ProgrammingLanguage programmingLanguage = plcBlock.ProgrammingLanguage;
    string blockAuthor = plcBlock.HeaderAuthor;
    string blockFamily = plcBlock.HeaderFamily;
    string blockTitle = plcBlock.HeaderName;
    string blockVersion = plcBlock.HeaderVersion;
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.5 Supprimer un bloc

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Code du programme

Pour supprimer un bloc, modifiez le code de programme suivant :

```
private static void DeleteBlocks(PlcSoftware plcsoftware)
//Runs through block group and deletes blocks
{
    PlcBlockSystemGroup group = plcsoftware.BlockGroup;
    //  or BlockUserGroup group = ...;
    for (int i = group.Blocks.Count - 1; i >= 0; i--)
    {
        PlcBlock block = group.Blocks[i];
        if (block != null)
        {
            block.Delete();
        }
    }
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.6 Supprimer un type de données utilisateur

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Code du programme

Pour supprimer un type utilisateur, modifiez le code de programme suivant :

```
private static void DeleteUserDataType(PlcSoftware plcSoftware)
{
    PlcTypeSystemGroup typeGroup = plcSoftware.TypeGroup;
    PlcTypeComposition dataTypes = typeGroup.Types;
    PlcType dataType = dataTypes.Find("DataTypeName");
    if (dataType != null)
    {
        dataType.Delete();
    }
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.7 Créer un groupe pour blocs

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour créer un groupe pour blocs, modifiez le code de programme suivant :

```
private static void CreateBlockGroup(PlcSoftware plcsoftware)
//Creates a block group
{
    PlcBlockSystemGroup systemGroup = plcsoftware.BlockGroup;
    PlcBlockUserGroupComposition groupComposition = systemGroup.Groups;
    PlcBlockUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.8 Supprimer un groupe pour blocs

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Code du programme

Pour supprimer un groupe pour blocs, modifiez le code de programme suivant :

```
// Deletes user groups from PlcBlockSystemGroup or PlcBlockUserGroup
private static void DeleteBlockFolder(PlcSoftware plcSoftware)
{
    PlcBlockUserGroup group = plcSoftware.BlockGroup.Groups.Find("myGroup");
    //PlcBlockSystemGroup group = plcSoftware.BlockGroup;
    PlcBlockUserGroupComposition subgroups = group.Groups;
    PlcBlockUserGroup subgroup = subgroups.Find("myUserGroup");
    if (subgroup != null)
    {
        subgroup.Delete();
    }
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.9 Interroger un groupe système pour blocs système

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme :

Pour déterminer le groupe créé par le système pour les blocs système, modifiez le code de programme suivant :

```
PlcSoftware plcSoftware = ...
foreach (PlcSystemBlockGroup systemGroup in plcSoftware.BlockGroup.SystemBlockGroups)
{
    foreach (PlcSystemBlockGroup group in systemGroup.Groups)
    {
        PlcBlockComposition pbComposition = group.Blocks;
        foreach (PlcBlock block in pbComposition)
        {
            //Ajoutez votre code ici
        }
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.10 Enumérer les sous-groupes système

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme : Enumérer tous les sous-groupes système

Pour énumérer les sous-groupes système de tous les blocs système, modifiez le code de programme suivant :

```
//Retrieves the system generated group for system blocks
private static void GetSystemgroupForSystemblocks(PlcSoftware plcSoftware)
{
    PlcSystemBlockGroupComposition systemBlockGroups =
plcSoftware.BlockGroup.SystemBlockGroups;
    if (systemBlockGroups.Count != 0)
    {
        PlcSystemBlockGroup sbSystemGroup = systemBlockGroups[0];
        foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
        {
            EnumerateSystemBlockGroups(group);
        }
    }
}
private static void EnumerateSystemBlockGroups(PlcSystemBlockGroup systemBlockGroup)
{
    foreach (PlcSystemBlockGroup group in systemBlockGroup.Groups)
    {
        // recursion EnumerateSystemBlockGroups(group);
    }
}
```

Code du programme : Accéder à un sous-groupe déterminé

Pour accéder à un sous-groupe déterminé, modifiez le code de programme suivant :

```
private static void AccessSbGroup(PlcSystemBlockGroup systemBlockGroup)
{
    PlcSystemBlockGroup group1 = systemBlockGroup.Groups.Find("User group XYZ");
    PlcSystemBlockGroup group2 = group1.Groups.Find("User group ZYX");
}
```

Voir aussi

Bibliothèques standard (Page 34)

Ajouter un fichier externe (Page 182)

7.12.1.11 Ajouter un fichier externe

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness :
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous pouvez ajouter un fichier externe à un API. Ce fichier externe est enregistré sous le chemin défini dans le système de fichiers.

Les formats suivants sont pris en charge :

- LIST
- SCL
- DB
- UDT

Remarque

L'accès à des groupes dans le dossier "Fichiers sources externes" n'est pas pris en charge.

Une exception se déclenche si vous indiquez une autre extension de fichier que *.AWL, *.SCL, *.DB ou *UDT.

Code de programme

Pour créer un fichier externe dans le dossier "Fichiers sources externes" à partir d'un bloc, modifiez le code de programme suivant :

```
private static void CreateBlockFromFile(PlcSoftware plcsoftware)
// Creates a block from a AWL, SCL, DB or UDT file
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.CreateFromFile("SomeBlockNameHere","SomePa
thHere");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.12 Générer une source à partir d'un bloc

Conditions requises

- L'application Openness est connectée à TIA Portal.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

L'interface API prend en charge la génération de sources à partir de blocs et de types de données utilisateur.

Pour les blocs, seuls les langages de programmation STL et SCL sont pris en charge. Des exceptions sont déclenchées dans les cas suivants :

- Le langage de programmation n'est pas STL ou SCL
- Un fichier du même nom existe déjà dans l'emplacement de sauvegarde cible.

Seule l'extension de fichier "*.udt" est prise en charge pour les types de données utilisateur. Des exceptions sont déclenchées dans les cas suivants :

- L'extension de fichier pour blocs de données n'est pas "*.db"
- L'extension de fichier pour blocs STL n'est pas "*.awl"
- L'extension de fichier pour blocs SCL n'est pas "*.scl"

Code du programme

Pour générer les fichiers sources à partir de blocs et de types, modifiez le code de programme suivant :

```
...
PlcBlock FC1 = ...;
PlcBlockSystemGroup plcBlockSystemGroup = ...;
// fails if programming language is not SCL
plcBlockSystemGroup.GenerateSourceFromBlocks(new PlcBlock[] { FC1 },
"SomePathHere.scl");
...
or
...
PlcBlock FC1 = ...;
PlcBlock FC2 = ...;
PlcBlockSystemGroup plcBlockSystemGroup = ...;
// fails if programming language is not SCL
plcBlockSystemGroup.GenerateSourceFromBlocks(new PlcBlock[] { FC1, FC2 },
"SomePathHere.scl");
...
or
... PlcBlock FC1 = ...;
PlcBlock FC2 = ...;
PlcBlockSystemGroup plcBlockSystemGroup = ...;
// fails if programming language is not SCL
plcBlockSystemGroup.GenerateSourceFromBlocks(new List<PlcBlock> { FC1,
FC2 }, "SomePathHere.scl");
...
or
...
IEnumerable<PlcBlock> blocks = ...;
PlcBlockSystemGroup plcBlockSystemGroup = ...;
// fails if programming language is not SCL
plcBlockSystemGroup.GenerateSourceFromBlocks(blocks, "SomePathHere.scl");
...
or
...
PlcBlockComposition blocks = ...;
PlcBlockSystemGroup plcBlockSystemGroup = ...;
// fails if programming language is not SCL
plcBlockSystemGroup.GenerateSourceFromBlocks(blocks, "SomePathHere.scl");
...
or
...
IEnumerable<PlcType> plcTypes = ...;
PlcTypeSystemGroup plcTypeSystemGroup = ...;
plcTypeSystemGroup.GenerateSourceFromTypes(plcTypes, "SomePathHere.scl");
...
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.1.13 Générer les blocs à partir de la source

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

Vous pouvez générer des blocs à partir de tous les fichiers externes du groupe "Fichiers sources externes". Seuls les fichiers externes au format ASCII sont pris en charge.

Remarque

L'accès à des groupes dans le dossier "Fichiers sources externes" n'est pas pris en charge.

Les blocs existants sont écrasés.

Si une erreur apparaît lors de l'appel, une Exception est déclenchée. Les 256 premiers caractères de tout message d'erreur sont compris dans le message de l'Exception. Le projet est remis à l'état de traitement où il se trouvait avant l'exécution de la méthode `GenerateBlocksFromSource`.

Code du programme

Pour générer les blocs à partir de tous les fichiers externes du groupe "Fichiers sources externes", modifiez le code de programme suivant.

```
// Creates a block from an external source file
PlcSoftware plcSoftware = ...;
foreach (PlcExternalSource plcExternalSource in
plcSoftware.ExternalSourceGroup.ExternalSources)
{
    plcExternalSource.GenerateBlocksFromSource();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.14 Supprimer un fichier externe

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Vous avez ouvert un projet par le biais d'une application Openness :
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Code du programme

Pour supprimer un fichier externe dans le groupe "Fichiers sources externes", modifiez le code de programme suivant :

Remarque

L'accès à des groupes dans "Fichiers sources externes" n'est pas pris en charge.

```
// Deletes an external source file
private static void DeleteExternalSource(PlcSoftware plcSoftware)
{
    PlcExternalSource externalSource =
plcSoftware.ExternalSourceGroup.ExternalSources.Find("myExternalsource");
    externalSource.Delete();
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.1.15 Démarrer un éditeur de bloc

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'instance du portail TIA est ouverte avec interface utilisateur.

Code du programme

Pour démarrer l'éditeur correspondant à une référence d'objet du type `PlcBlock` dans l'instance de TIA Portal, modifiez le code de programme suivant :

```
//Opens a block in a block editor
private static void StartBlockEditor(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.ShowInEditor();
}
```

Pour ouvrir l'éditeur correspondant à une référence d'objet du type `PlcType` dans l'instance de TIA Portal, modifiez le code de programme suivant :

```
//Opens a udt in udt editor
private static void StartPlcTypEditor(PlcSoftware plcSoftware)
{
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find("my_udt");
    udt.ShowInEditor();
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.2 Tables des variables

7.12.2.1 Créer les groupes personnalisés pour variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface API prend en charge la création de groupes personnalisés pour variables API.

Code du programme

Pour créer un groupe personnalisé pour variables API, modifiez le code de programme suivant :

```
//Creates a plc tag table user group
private static void CreatePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup systemGroup = plcSoftware.TagTableGroup;
    PlcTagTableUserGroupCompostion groupComposition = systemGroup.Groups;
    PlcTagTableUserGroup myCreatedGroup = groupComposition.Create("MySubGroupName");
    // Optional;
    // create a subgroup
    PlcTagTableUserGroup mySubCreatedGroup =
myCreatedGroup.Groups.Create("MySubSubGroupName");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.2.2 Supprimer les groupes personnalisés pour variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface API prend en charge la suppression d'un groupe personnalisé déterminé pour tables de variables API.

Code du programme

Pour supprimer un groupe personnalisé déterminé pour tables de variables API, modifiez le code de programme suivant :

```
private static void DeletePlcTagTableUserGroup(PlcSoftware plcSoftware)
{
    PlcTagTableUserGroup group = plcSoftware.TagTableGroup.Groups.Find("MySubGroupName");
    if (group != null)
    {
        group.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.2.3 Supprimer la table des variables API dans un groupe

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour supprimer une table des variables déterminée dans un groupe, modifiez le code de programme suivant :

```
//Deletes a PlcTagTable of a group
private static void DeletePlcTagTableInAGroup(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup group = plcSoftware.TagTableGroup;
    PlcTagTable tagtable = group.TagTables.Find("MyTagTable");
    if (tagtable!= null)
    {
        tagtable.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.2.4 Supprimer une variable individuelle d'une table des variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code du programme

Pour supprimer une variable déterminée dans une table des variables API, modifiez le code de programme suivant :

```
private static void DeleteTagFromPLCTagtable(PlcSoftware plcsoftware)
// Deletes a single tag of a tag table
{
    string tagName = "MyTag";
    PlcTagTable table = plcsoftware.TagTableGroup.TagTables.Find("myTagTable");
    PlcTagComposition tagComposition = table.Tags;
    PlcTag tag = tagComposition.Find(tagName);
    if (tag != null)
    {
        tag.Delete();
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.2.5 Démarrer l'éditeur "Variables"

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'instance du portail TIA est ouverte avec interface utilisateur.

Code du programme

Pour démarrer l'éditeur correspondant à une référence d'objet du type `PlcTagTable` dans l'instance de TIA Portal, modifiez le code de programme suivant :

```
//Ouvre la table des variables dans l'éditeur "Tags"
private static void OpenTagtableInEditor(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
plcTagTable.ShowInEditor();
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

7.12.2.6 Lire la date et l'heure de la dernière modification d'une table de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Le format de l'horodatage est l'UTC.

Code du programme

Pour lire l'horodatage d'une table de variables API déterminée, modifiez le code de programme suivant :

```
//Reads Time-Stamp of a plc Tag Table
private static void GetLastModificationDateOfTagtable(PlcSoftware plcSoftware)
{
    PlcTagTable plcTagTable = plcSoftware.TagTableGroup.TagTables.Find("MyTagTable");
    DateTime modifiedTagTableTimeStamp = plcTagTable.ModifiedTimeStamp;
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.12.3 Supprimer un groupe personnalisé dans un appareil API

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Code de programme

Pour supprimer un groupe personnalisé dans un appareil API, modifiez le code de programme suivant :

```
PlcSoftware plcSoftware = ...;
PlcBlockUserGroup blockUserGroup = plcSoftware.BlockGroup.Groups.Find("MyUserFolder");
blockUserGroup.Delete();
```

7.13 Concepts de base

7.13.1 Traitement des exceptions

Exceptions en cas d'accès au portail TIA avec des API publics

Lors de l'exécution d'une application Openness avec l'API public, toutes les erreurs qui se sont produites sont signalées comme des exceptions. Ces exceptions contiennent des informations qui vous aident à éliminer les erreurs survenues.

Il existe deux types d'exception :

- Recoverable (Siemens.Engineering.EngineeringException)
  Avec cette exception, vous continuez d'accéder à TIA Portal sans interruption. Vous pouvez également couper la liaison au portail TIA.
  Les EngineeringExceptions contiennent les types suivants :
  - Exceptions de sécurité (EngineeringSecurityException), par exemple en cas d'absence de droits d'accès.
  - Exceptions lors de l'accès aux objets (EngineeringObjectDisposedException), par ex. lors de l'accès à des objets qui n'existent plus.
  - Exceptions lors de l'accès aux attribut (EngineeringNotSupportedException), par ex. lors de l'accès à des attributs qui n'existent plus.
  - Exceptions générales lors de l'appel (EngineeringTargetInvocationException), par ex. en cas d'erreur malgré des appels valides de l'API public.
  - Exceptions lors de l'appel (EngineeringRuntimeException), par ex. lors d'une affectation invalide.
  - Exceptions lorsque les appels sont terminés (EngineeringUserAbortException), par ex. lors de l'interruption du processus d'importation par l'utilisateur.

  Les EngineeringExceptions ont les attributs suivants :
  - `ExceptionMessageData messageData`: Contient la cause pour laquelle l'exception a été déclenchée.
  - `ExceptionMessageData detailMessageData`: Contient des informations supplémentaires sur la cause. Le résultat est fourni en retour sous forme de <IList>.
  - `String message` : Fournit en retour le résultat issu de `MessageData` et de `DetailMessageData`.

  `ExceptionMessageData` fournit en retour les informations suivantes :
  - `String Text` : Contient la cause pour laquelle l'exception a été déclenchée.
  - `Int ServiceId` : Fournit l'ID du service ayant déclenché l'exception.
  - `Int MessageId` : ID univoque au sein du service.
- NonRecoverable (Siemens.Engineering.NonRecoverableException)
  Dans le cas de cette exception, le portail TIA est fermé et la liaison au portail TIA est coupée. Vous devez redémarrer le portail TIA avec l'application Openness.

Code du programme

L'exemple suivant montre les possibilités que vous avez pour réagir à des exceptions :

```
try
{
    ...
}

catch(EngineeringSecurityException engineeringSecurityException)
{
    Console.WriteLine(engineeringSecurityException);
}

catch(EngineeringObjectDisposedException engineeringObjectDisposedException)
{
    Console.WriteLine(engineeringObjectDisposedException.Message);
}

catch(EngineeringNotSupportedException engineeringNotSupportedException)
{
    Console.WriteLine(engineeringNotSupportedException.MessageData.Text);
    Console.WriteLine();
    Foreach(ExceptionMessageData detailMessageData in
engineeringNotSupportedException.DetailMessageData)
    {
        Console.WriteLine(detailMessageData.Text);
    }
}

catch (TargetInvocationException)
{
    throw;
}

catch (EngineeringException)
{
    //Do not catch general exceptions
    throw;
}

catch(NonRecoverableException nonRecoverableException)
{
    Console.WriteLine(nonRecoverableException.Message);
}
```

7.13.2 Utilisation d'associations

Accéder aux affectations

Une affectation décrit la relation entre deux objets ou plus au niveau du type.

TIA Portal Openness V14 prend en charge l'accès aux affectations via l'index et les boucles foreach. L'accès direct, par exemple via string name, n'est pas pris en charge.

Propriétés

Les propriétés suivantes sont disponibles :

- `int Count`
- `bool IsReadonly`
- `IEngineeringObject Parent`
- `retType this [ int index ] { get; }`

Méthodes

TIA Portal Openness V14 prend en charge les méthodes suivantes :

- `int IndexOf ( type )` : fournit en retour l'index dans l'affectation pour une instance transmise :
- `bool Contains ( type )` : détermine si l'instance transmise est contenue dans l'affectation.
- `IEnumerator GetEnumerator <retType>()` : est utilisé dans des boucles foreach et permet d'accéder à un objet.
- `void Add ( type )`[1] : ajoute l'instance transmise de l'affectation.
- `void Remove ( type )`[1] : supprime l'instance transmise de l'affectation.

[1] : n'est pas prise en charge par toutes les affectations.

7.13.3 Utilisation de compositions

Appel de compositions

Une composition est un cas particulier d'affectation Une composition exprime une relation sémantique entre deux objets dont l'un est une partie de l'autre.

Propriétés

Les propriétés suivantes sont disponibles :

- `int Count`
- `bool IsReadonly`
- `IEngineeringObject Parent`
- `retType this [int index] {get;}` : accède de manière indexée à un objet de la composition.
  Vous devez utiliser ce type d'accès de manière ciblée uniquement, car chaque accès indexé dépasse les limites du processus.

Méthodes

TIA Portal Openness V14 prend en charge les méthodes suivantes :

- `retType Create (id, …)` : crée une nouvelle instance et l'ajoute à la composition. La signature de la méthode dépend du type de création de l'instance. Cette méthode n'est pas prise en charge par toutes les compositions.
- `type Find (id, …)` : Recherche dans une composition l'instance ayant l'ID transmis. La recherche n'est pas récursive. La signature de la méthode dépend du type de recherche de l'instance. Cette méthode n'est pas prise en charge par toutes les compositions.
- `IEnumerator GetEnumerator<retType> ()` : est utilisé dans des boucles foreach et permet d'accéder à un objet.
- `Delete (type)`[1] : supprime l'instance spécifiée par la référence d'objet actuelle.
- `int IndexOf (type)` : fournit en retour l'index dans la composition pour une instance transmise.
- `bool Contains (type)` : détermine si l'instance transmise est contenue dans la composition.
- `void Import(string path, ImportOptions importOptions)`[1] : S'applique à chaque composition comportant des types pouvant être importés.
  Chaque signature d'importation contient un paramètre de configuration du type "ImportOptions (Page 211)" ("None", "Overwrite") qui permet à l'utilisateur de commander le comportement à l'importation.

[1] : Pas prise en charge par toutes les compositions.

7.13.4 Vérifier l'égalité des objets

Utilisation

En tant qu'utilisateur d'un API public, vous pouvez vérifier l'identité des objets à l'aide du programme :

- Vous vérifiez alors avec l'opérateur `==` si deux références d'objet sont identiques.
- La méthode `System.Object.Equals()` vous permet de vérifier si deux objets sont réellement identiques en ce qui concerne le portail TIA.

Code du programme

Pour vérifier les types de référence d'objet, modifiez le code de programme suivant :

```
...
//Composition
IDeviceComposition sameCompA = project.Devices;
IDeviceComposition sameCompB = project.Devices;
if (sameCompA.Equals(sameCompB))
{
    Console.WriteLine("sameCompA is equal to sameCompB");
}
if (!(sameCompA == sameCompB))
{
    Console.WriteLine("sameCompA is not reference equal to sameCompB");
}
IDeviceComposition sameCompAsA = sameCompA;
if (sameCompAsA.Equals(sameCompA))
{
    Console.WriteLine("sameCompAsA is equal to sameCompA");
}
if (sameCompAsA == sameCompA)
{
    Console.WriteLine("sameCompAsA is reference equal to sameCompA");
}
MultiLingualGraphicComposition notSameComp = project.Graphics;
if (!sameCompA.Equals(notSameComp))
{
    Console.WriteLine("sameCompA is not equal to notSameComp");
}
```

Voir aussi

Bibliothèques standard (Page 34)

7.13.5 Opérations de lecture pour attributs

Opérations d'ensemble et opérations de lecture standards pour attributs

TIA Portal Openness V14 prend en charge l'accès aux attributs par les méthodes suivantes, disponibles au niveau de l'objet :

- Opérations d'ensemble pour l'accès en lecture
- Opérations standard de lecture

Code du programme pour les opérations d'ensemble

```
//Exercise GetAttributes and GetAttributeNames
//get all available attributes for a device,
//then get the names for those attributes, then display the results.
private static void DynamicTest(Project project)
{
    IDevice device = project.Devices[0];
    IList<string> attributeNames = new List<string>();
    IList<EngineeringAttributeInfo> attributes =
((IEngineeringObject)device).GetAttributeInfos();
    foreach (EngineeringAttributeInfo engineeringAttributeInfo in attributes)
    {
        string name = engineeringAttributeInfo.Name;
        attributeNames.Add(name);
    }
    IList<object> values = ((IEngineeringObject)device).GetAttributes(attributeNames);
    for (int i = 0; i < attributes.Count; i++)
    {
        Console.WriteLine("attribute name: " + attributeNames[i] + " value: " + values[i]);
    }
}
```

Opérations d'ensemble pour l'accès en lecture

Cette méthode est disponible pour chaque objet :

```
public abstract IList<object> GetAttributes(IList<string> names);
```

Opérations standard de lecture

Les opérations standard de lecture sont **des méthodes optionnelles**. Utilisez ces méthodes sur des objets qui prennent en charge des attributs dynamiques.

Les options suivantes sont disponibles :

- Méthode standard pour tous les noms d'attributs actuellement disponibles, en fonction de l'état actuel de l'objet
  `public abstract IList<string> GetAttributeNames();`
- Méthode standard pour la lecture d'un attribut
  `public abstract object GetAttribute(string name);`

Remarque

Les attributs dynamiques ne s'affichent pas dans IntelliSense car leur disponibilité dépend de l'état de l'instance d'objet.

Exportation/importation 8

8.1 Vue d'ensemble

8.1.1 Notions élémentaires sur l'importation/exportation

Introduction

Vous pouvez exporter certaines données de configuration puis les réimporter après édition, soit dans le même projet, soit dans un autre.

Remarque

L'utilisation de cette description pour éditer et exploiter manuellement le fichier source n'entraîne aucune obligation ni garantie d'aucune sorte. Siemens décline donc toute responsabilité en cas d'utilisation de cette description ou de parties de cette description.

Objets exportables et importables

Vous pouvez également importer ou exporter les données de configuration suivantes par le biais de Public API :

Tableau 8-1 Projets

Objets	Exportation	Importation
Bibliothèque de graphiques	X	X

Tableau 8-2 API

Objets	Exportation	Importation
Blocs	X	X
Blocs avec protection Know How	X	–
Blocs F	X	–
Blocs système	X	–
Tables de variables API	X	X
Variables API	X	X
Types de données utilisateur	X	X

Tableau 8-3 IHM

Objets	Exportation	Importation
Vues	X	X
Modèles de vue	X	X
Vues globales	X	X
Scripts	X	X
Listes de textes	X	X
Listes de graphiques	X	X
Cycles	X	X
Connexions	X	X
Table des variables	X	X
Variables	X	X

Exportation complète ou de références ouvertes

Les types d'objets dont la liste est dressée ci-dessus sont exportés ou importés avec tous les objets lorsqu'ils appartiennent à la même arborescence. Cela vaut également pour les objets référencés de la même arborescence.

Les objets référencés dans d'autres arborescences ne peuvent pas, quant à eux contraire, être complètement exportés ou importés. Des "références ouvertes" à ces objets sont exportées ou importées à leur place.

Les objets référencés de la même arborescence sont exportés uniquement s'ils font partie du groupe des objets exportables. Toutes les dynamisations s'appliquant à des objets sont traitées comme des objets lors de l'importation/exportation et sont également exportées et importées.

Lors de l'exportation, toutes les propriétés d'objet qui ont été modifiées durant la configuration sont exportées. Cela s'applique toujours, qu'une propriété modifiée soit utilisée ou non.

Exemple : Vous avez configuré un champ d'E/S graphique avec le mode "Entrée/Sortie" et sélectionné le réglage "Visible après avoir cliqué" pour la propriété "Barre de défilement". Puis vous avez basculé le mode sur "Deux états" pendant la configuration. Dans ce mode, la propriété "Barre de défilement" n'est pas disponible. Etant donné que la propriété "Barre de défilement" a été modifiée, elle est exportée lors de l'exportation, bien qu'elle ne soit pas utilisée.

Importation de références ouvertes

Vous pouvez également importer des objets assortis de références ouvertes (voir Importation de données de configuration (Page 211)).

Si les objets référencés se situent dans le projet cible, les références ouvertes sont automatiquement liées à nouveau aux types d'objet. Ces objets doivent se situer au même endroit et porter le même nom pendant l'exportation. Si les objets référencés ne sont pas situés dans le projet cible, les références ouvertes ne peuvent pas être résolues. Aucun objet supplémentaire n'est créé pour la résolution des références ouvertes.

Importation et exportation de polices de caractère

Les polices définies pour des objets sont également exportées et importées.

Si vous importez des polices qui ne sont pas incluses dans le projet, la police par défaut s'affiche pour l'objet après l'importation. La police importée est toutefois enregistrée dans la gestion des données.

Si les attributs d'une police ne sont pas définis dans le fichier d'importation, les attributs sont dotés de valeurs par défaut après l'importation.

Restrictions

Le format d'exportation est interne et n'est valable que pour la version V14. Le format d'exportation peut être modifié pour les versions ultérieures.

Toutes les erreurs survenant au cours de l'importation ou de l'exportation sont signalées comme des exceptions.
Pour plus d'informations sur les exceptions, veuillez vous référer au chapitre Traitement des exceptions (Page 193).

Voir aussi

Domaine d'utilisation de l'importation/exportation (Page 201)

Exportation de données de configuration (Page 210)

Structure d'un fichier XML (Page 203)

8.1.2 Domaine d'utilisation de l'importation/exportation

Introduction

La fonction d'importation/exportation vous permet d'exporter certains objets ou groupes d'objets de manière ciblée.

Vous pouvez éditer les données exportées avec un programme externe ou les réutiliser telles quelles dans d'autres projets TIA Portal.

Si vous structurez correctement le fichier d'importation, vous pouvez également importer sans exportation préalable des données de configuration créées en externe.

Remarque

L'importation de données de configuration créées en externe avec des erreurs de code ou de structure erronée peut provoquer des erreurs inattendues.

Domaine d'application

Exporter et importer des données est utile pour les tâches suivantes :

- éditer des données de configuration en externe,
- importer des données de configuration générées en externe, telles que des listes de textes et des variables,
- distribuer à différents projets des données de configuration prédéfinies, p. ex. une vue de processus modifiée qui doit être utilisée dans plusieurs projets.

Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

8.1.3 Importation SimaticML spécifique à la version

Utilisation

L'importation SimaticML s'effectue indépendamment de la version à partir de Openness V14. Vous pouvez désormais utiliser des objets de données exportés de versions actuelles dans des versions ultérieures. Le système utilise la version d'Openness spécifiée ppur la création d'un objet.

Remarque

Des objets après V14 ne peuvent pas être importés dans un TIA Portal V14, mais des objets V14 peuvent être importés dans un TIA Portal à partir de V14.

Pour prendre en charge cette caractéristique, les fichiers SimaticML contiennent maintenant les informations de version de modèle représentées ci-dessous :

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
    <Engineering version="V14"/>
    <DocumentInfo>
     ...
    </DocumentInfo>
    <SW.DataBlock ID="0">
     ...
    </SW.DataBlock>
</Document>
```

Remarque

Si ces informations de version ne sont pas présentes dans le fichier SimaticML, le système utilise la version de modèle actuelle.

8.1.4 Structure d'un fichier XML

Introduction

Les données du fichier d'exportation issues de l'importation/exportation sont organisées au moyen d'une structure de base.

Structure de base d'un fichier d'exportation

Le fichier d'exportation est créé au format XML.

Le fichier XML commence par des informations sur le document. Il comporte les données de l'installation spécifique à l'ordinateur avec laquelle le projet a été exporté.

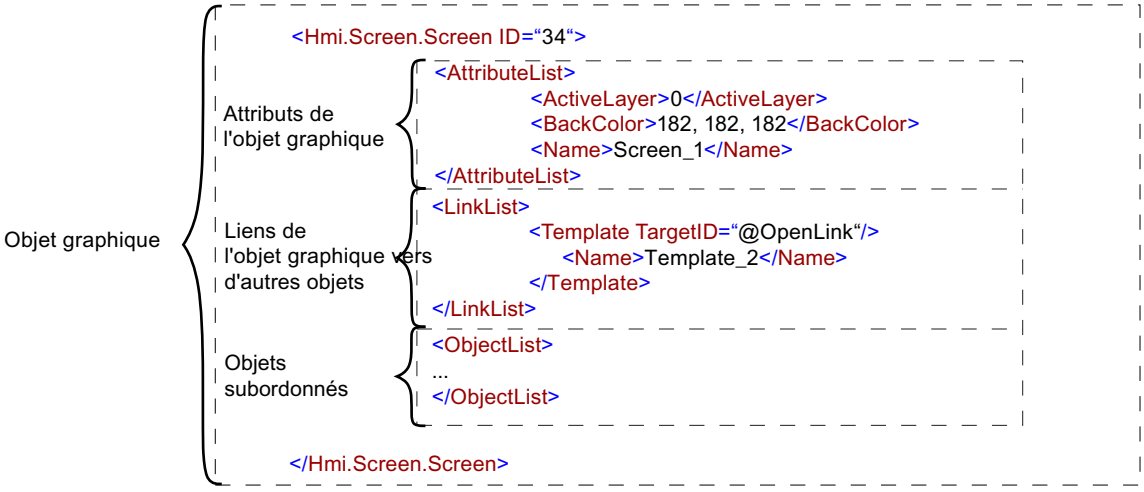
Le fichier d'exportation comprend les deux zones suivantes :

- Informations sur le document
  Cette zone vous permet d'indiquer vos propres informations relatives à l'exportation et ce dans une syntaxe XML valide. L'importation ignore le contenu.
  Vous pouvez par ex. insérer un bloc `<IntegrityInformation>...</IntegrityInformation>` en plaçant des informations supplémentaires à la validation. Après la transmission du fichier XML, le destinataire peut vérifier avant l'importation avec ce bloc si le fichier XML a été modifié.
- Objet
  Cette zone contient les éléments à exporter.


Informations sur le document
Objet graphique
<?xml version="1.0" encoding="UTF-8" ?>
<Document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<DocumentInfo>
<UserName>Jane Doe</UserName>
<Company>Example_Inc</Company>
<IntegrityInformation>...</IntegrityInformation>
<Created>2016-04-28T18:05:42.179207Z</Created>
<ExportSetting>WithDefaults</ExportSetting>
<InstalledProducts>
<Product>
<DisplayName>Totally Integrated Automation Portal</DisplayName>
<DisplayVersion>V14</DisplayVersion>
</Product>
<OptionPackage>
<DisplayName>WinCC Professional</DisplayName>
<DisplayVersion>V14</DisplayVersion>
</OptionPackage>
<OptionPackage>
<DisplayName>Siemens TIA Openness</DisplayName>
<DisplayVersion>V14</DisplayVersion>
</OptionPackage>
</InstalledProducts>
</DocumentInfo>
<Hmi.Screen.Screen ID ="0">
<AttributeList>
<ActiveLayer>0</ActiveLayer>
<BackColor>189,190,0</BackColor>
<Height>422</Height>
<Name>Root screen</Name>
<Number>1</Number>
<Visible>True</Visible>
<Width>640</Width>
</AttributeList>
<LinkList>
<Template TargetID="@OpenLink">
<Name>Template_1</Name>
</Template>
</LinkList>
<ObjectList>
<Name>dummy</Name>
</ObjectList>
</Hmi.Screen.Screen>
</Document>

Objets graphiques d'un fichier d'exportation

Les éléments exportés sont disponibles dans d'autres éléments du fichier XML.



Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

8.1.5 Structure des données pour l'importation/exportation

Objets

La structure de base est la même pour tous les objets.

Chaque objet du fichier XML débute par son type, p. ex. "Hmi.Screen.Button" et un ID. L'ID est automatiquement générée durant l'exportation.

```
<Hmi.Screen.Button Name="Button_1" CompositionName="ScreenItems" ID="60">
```

Excepté l'objet de départ, chaque objet contient également un attribut XML "CompositionName". La valeur de cet attribut est prédéfinie. Dans quelques cas, vous devez spécifier cet attribut, p. ex. pour changer d'inscription quand un bouton est enfoncé ou relâché.

```
<MultilingualText ID="8" CompositionName="TextOff">
    <AttributeList>
        <TextItems>
            <Value lang="en-US">
                <body>
                    <p>TextOff</p>
                </body>
            </Value>
        </TextItems>
    </AttributeList>
</MultilingualText>
<MultilingualText ID="9" CompositionName="TextOn">
    <AttributeList>
        <TextItems>
            <Value lang="en-US">
                <body>
                    <p>TextOn</p>
                </body>
            </Value>
        </TextItems>
    </AttributeList>
</MultilingualText>
```

Attributs

Chaque objet comprend des attributs qui sont contenus dans une section appelée "AttributeList". Chaque attribut est structuré comme élément XML, p. ex. "BackColor". La valeur d'un attribut est structurée comme contenu XML, p. ex "204, 204, 204".

```
<Hmi.Screen.Button ID="2" CompositionName="ScreenItems">
    <AttributeList>
        <BackColor>204, 204, 204</BackColor>
        <ObjectName>Button_1</ObjectName>
    </AttributeList>
</Hmi.Screen.Button>
```

Pour référencer des objets, chaque objet reçoit au besoin une section appelée "LinkList". Cette section contient des liaisons à d'autres objets à l'intérieur ou à l'extérieur du fichier XML. Chaque liaison est structurée comme élément XML. La désignation d'une liaison est prédéfinie par l'objet cible dans le fichier modèle. Chaque liaison comprend également l'attribut "TargetID". Si l'objet cible se trouve dans le fichier XML, la valeur de l'attribut "TargetID" est l'ID de l'objet référencé, précédé de dièse "#". Si l'objet cible ne se trouve pas dans le fichier XML, la valeur de l'attribut "TargetID" est égale à "@OpenLink". La référence à l'objet proprement dite est structurée comme un élément XML subordonné.

```
<Hmi.Tag.Tag ID="17">
    <AttributeList>
        <Name>Tag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>2 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_connection</Name>
        </Connection>
    </LinkList>
</Hmi.Tag.Tag>
```

Corrélation entre les objets et la structure XML

Les figures ci-dessous montrent la corrélation entre la structure XML exportée et les objets correspondants dans WinCC.

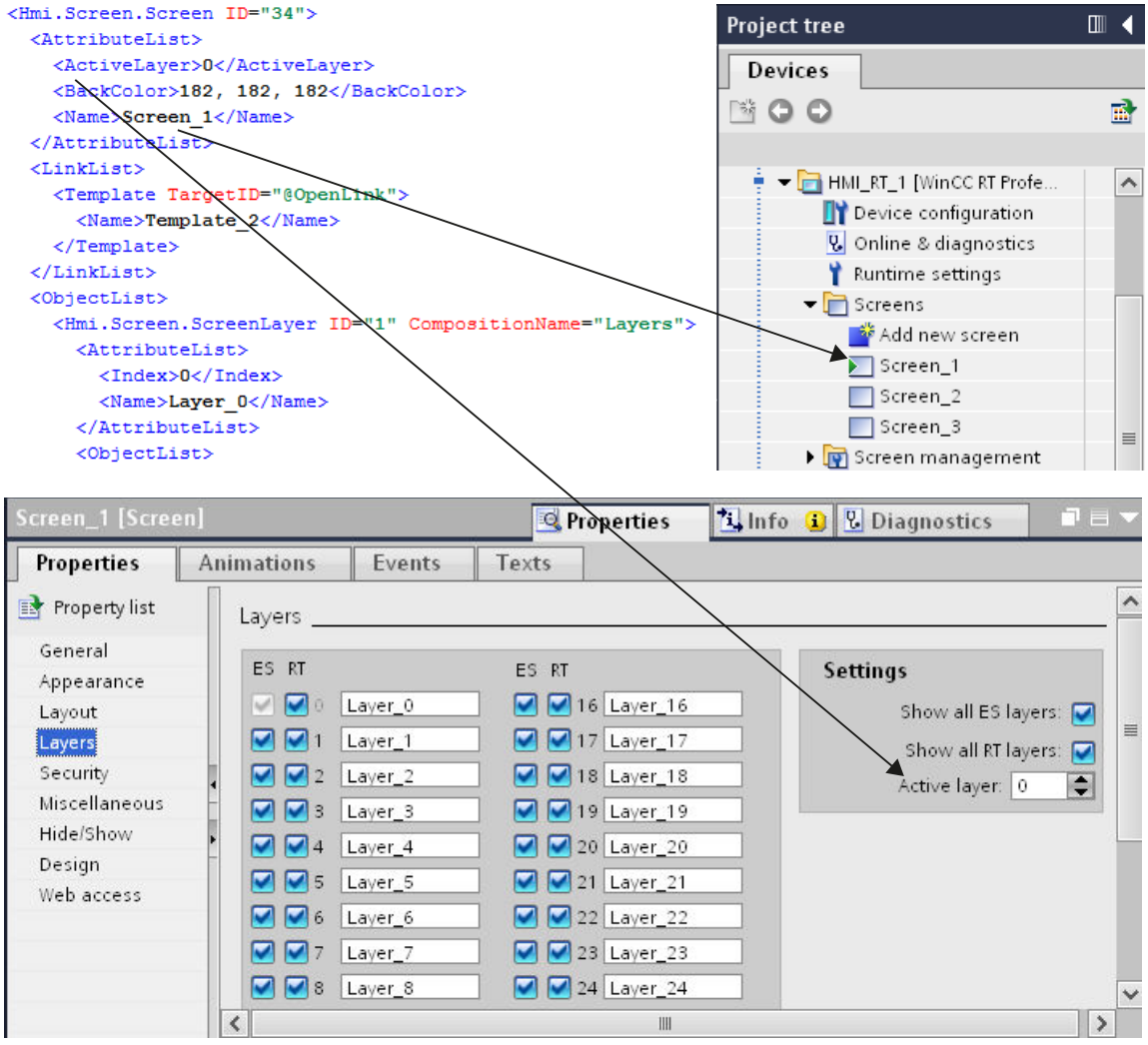


Figure 8-1 Corrélation entre l'interface utilisateur WinCC et la structure XML

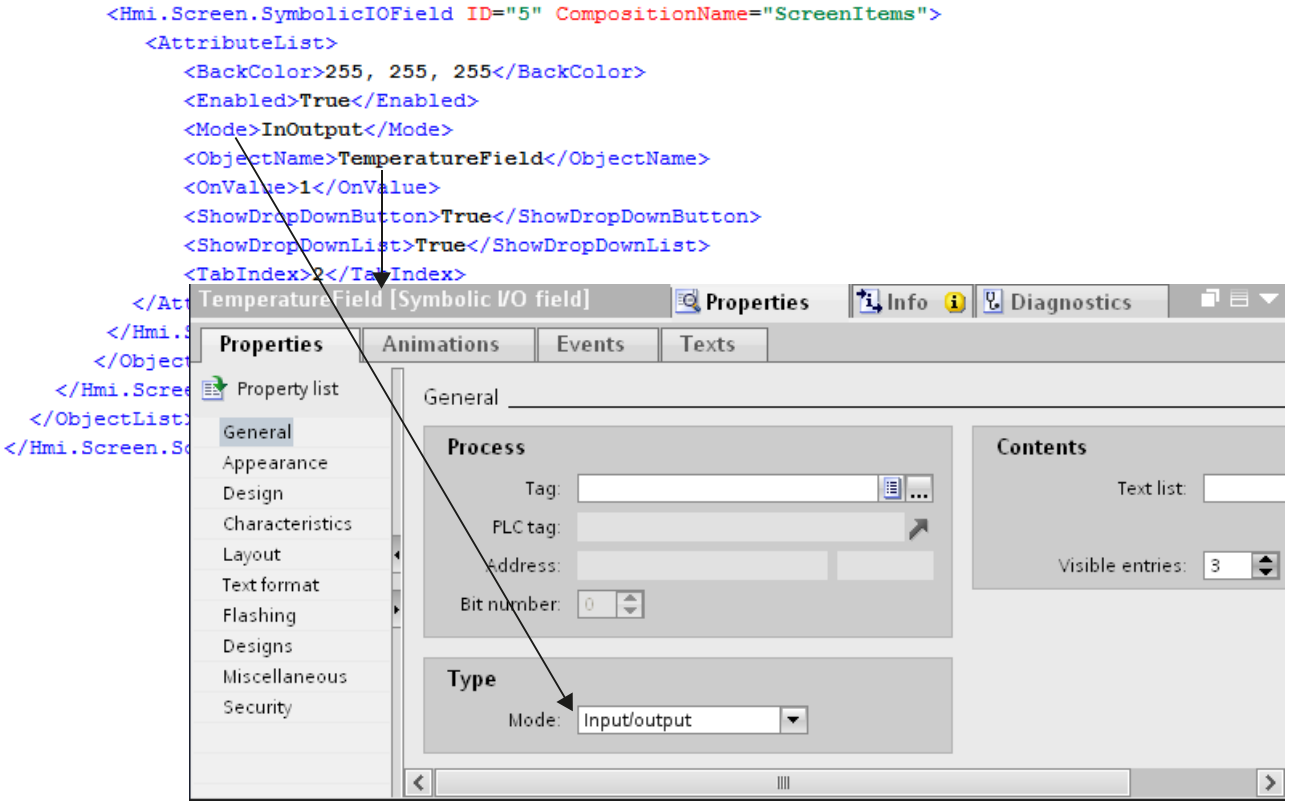


Figure 8-2 Corrélation entre les paramètres de WinCC et la structure XML

8.1.6 Edition du fichier XML

Introduction

Pour éditer un fichier XML destiné à l'importation de données de configuration, vous utilisez un éditeur XML ou un éditeur de texte.

Si vous effectuez des modifications importantes ou si vous créez vous-même des structures d'objet, il est recommandé d'utiliser un éditeur XML disposant d'une fonction de complément automatique.

Remarque

La modification du contenu XML requiert de solides connaissances de la structure et des règles de validation dans XML. Evitez les erreurs de validation et ne modifiez manuellement la structure XML qu'exceptionnellement.

8.1.7 Exportation de données de configuration

Introduction

Les données de configuration sont à chaque fois exportées dans un fichier XML par objet de départ (racine).

L'édition du fichier d'exportation requiert des connaissances en XML. Pour une édition simplifiée, utilisez un éditeur XML.

Exemple

Vous avez une vue de processus qui contient un champ E/S. Une variable externe est configurée pour ce champ E/S. Si vous exportez la vue de processus, la vue et le champ E/S sont exportés. La variable et la liaison utilisée par la variable ne sont pas exportées, seule une référence ouverte est exportée.

Contenu du fichier d'exportation

Dans le fichier d'exportation sont stockés tous les objets d'une arborescence, à partir de l'objet de départ, ainsi que leurs propriétés. En revanche, toutes les références aux objets d'autres arborescences sont exportés comme références ouvertes uniquement. Les propriétés correspondantes des objets référencés dans différentes arborescences ne sont pas écrites dans le fichier d'exportation.

Remarque

L'exportation de types d'objet de la bibliothèque n'est pas prise en charge.

Vous pouvez créer des objets comme type dans la bibliothèque. Les instances du type d'objet utilisées dans le projet peuvent être éditées avec l'application Openness comme d'autres objets. Si vous exportez des objets, les instances sont exportées sans les informations de type.

Si vous réimportez ces objets dans le projet, les instances des types d'objet sont écrasées et l'instance est coupée du type d'objet.

Le fichier d'exportation ne contient pas nécessairement tous les attributs d'un objet. C'est vous qui définissez les données à exporter :

- `ExportOptions.None`
  Ce paramétrage n'exporte que les données modifiées ou différentes des données standard. Le fichier d'exportation contient, de plus, toutes les valeurs obligatoires pour une importation ultérieure des données.
- `ExportOptions.WithDefaults`[1]
  De plus, les valeurs par défaut sont exportées.
- `ExportOptions.WithReadOnly`[1]
  De plus, les valeurs protégées en écriture sont exportées.

[1] : vous pouvez combiner ces deux options avec la syntaxe suivante :

```
Export(path,ExportOptions.WithDefaults |
ExportOptions.WithReadOnly);
```

Le contenu du fichier d'exportation est entièrement en anglais. Indépendamment de cela, les textes de projet sont exportés et importés dans toutes les langues disponibles.

Dans le fichier d'exportation, les données de configuration sont toutes structurées comme objets XML.

Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

Bibliothèques standard (Page 34)

Exporter des blocs (Page 257)

8.1.8 Importation de données de configuration

Introduction

Les données de configuration sont importées depuis un fichier XML exporté au préalable et édité ou bien depuis un fichier XML que vous créez vous-même. Les données contenues dans ce fichier sont contrôlées lors de l'importation. Cela garantit que l'importation ne provoquera pas une incohérence des données de configuration dans TIA Portal.

Restrictions

- Lors de l'importation de textes, les langues du projet correspondantes doivent être paramétrées dans le projet cible pour éviter que l'importation n'échoue.
- Seul les pointeurs de zone sous "separately for each connection" peuvent être importés ou exportés.
- Si plusieurs objets racine sont indiqués dans un fichier d'importation et que l'un de ces objets n'est pas valide, le contenu entier du fichier d'importation n'est pas importé.
- Si vous indiquez, dans le fichier d'importation, des propriétés d'un objet invalides, non éditables dans l'interface utilisateur graphique de TIA Portal, l'importation est annulée.

Remarque

Plages de valeurs pour les propriétés graphiques en fonction de l'appareil

Si les valeurs de propriétés graphiques se situent en dehors de la plage de valeurs valide, ces valeurs sont remises aux valeurs maximales possibles pour l'appareil IHM lors de l'importation.

Remarque

L'importation de types d'objet de la bibliothèque n'est pas prise en charge.

Vous pouvez créer des objets comme type dans la bibliothèque. Les instances du type d'objet utilisées dans le projet peuvent être éditées avec l'application Openness comme d'autres objets. Si vous exportez des objets, les instances sont exportées sans les informations de type.

Si vous réimportez ces objets dans le projet, les instances des types d'objet sont écrasées et l'instance est coupée du type d'objet.

Comportement d'importation différent

Si les objets à importer existent déjà dans le projet, vous devez commander le comportement d'importation à l'aide de différents codes de programme. Faute de quoi, les objets sont de nouveau créés dans le projet lors de l'importation.

Les paramétrages suivants peuvent être effectués pour définir le comportement d'importation :

- `ImportOptions.None`
  Ce paramètre permet d'importer les données de configuration sans écrasement.
  Si un objet existe déjà dans le projet lors de l'importation depuis un fichier XML, le processus est annulé par une exception.
- `ImportOptions.Override`
  Ce paramètre est utilisé pour l'importation des données de configuration avec écrasement automatique.
  Vous pouvez décider d'écraser les objets existants au sein du projet pendant l'importation. Les objets pertinents sont supprimés du projet avant l'importation et recréés avec des valeurs par défaut. Lors de l'importation, ces valeurs par défaut sont écrasées par des valeurs issues de l'importation.

Marche à suivre pour l'importation

Pour importer un fichier XML, il faut que les données qu'il contient satisfassent à certaines règles. Le contenu du fichier d'importation doit avoir la forme correcte. Il ne doit présenter aucune erreur de syntaxe ni aucune erreur dans la structure des données. En cas de modifications importantes, utilisez un éditeur XML, car ce dernier contrôle ces critères avant l'importation.

Lors de l'importation du fichier XML dans TIA Portal, les données contenues dans le fichier sont d'abord contrôlées afin d'exclure toute erreur formelle dans le code XML. Si des erreurs sont détectées lors de la vérification, l'importation est interrompue et les erreurs s'affichent dans une exception (voir Traitement des exceptions (Page 193)).

Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

Bibliothèques standard (Page 34)

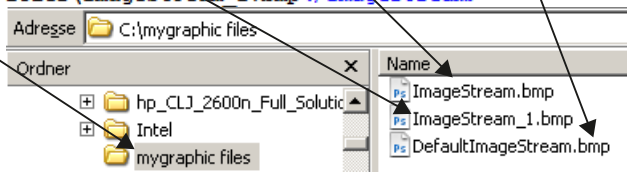
Importer un type de données utilisateur (Page 269)

8.1.9 Exportation/importation de graphiques

Introduction

Lorsque vous exportez des données de configuration à partir du portail TIA, les graphiques sélectionnés ou référencés par un objet ne sont pas enregistrés dans le fichier XML. Ils sont enregistrés séparément lors de l'exportation. Les graphiques sont référencés dans le fichier XML par une indication d'un chemin d'accès relatif et de leur nom de fichier. Dans le fichier XML, une référence à un graphique est structurée comme objet et contient, comme les autres objets, une liste d'attributs ainsi qu'une liste de liens le cas échéant.

```
<Hmi.Globalization.MultiLingualGraphic ID="0">
    <AttributeList>
      <DefaultDithering>False</DefaultDithering>
      <DefaultImageStream external="path">mygraphic files\DefaultImageStream.bmp</DefaultImageStream>
      <DefaultSmoothness>False</DefaultSmoothness>
      <Name>MyGraphic1</Name>
    </AttributeList>
    <ObjectList>
      <Hmi.Globalization.GraphicItem ID="1" CompositionName="Items">
        <AttributeList>
          <Culture>en-US</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
      <Hmi.Globalization.GraphicItem ID="2" CompositionName="Items">
        <AttributeList>
          <Culture>de-DE</Culture>
          <Dithering>False</Dithering>
          <ImageStream external="path">mygraphic files\ImageStream_1.bmp</ImageStream>
          <Smoothness>False</Smoothness>
        </AttributeList>
      </Hmi.Globalization.GraphicItem>
    </ObjectList>
  </Hmi.Globalization.MultiLingualGraphic>
```



Exportation de graphiques

Lors de l'exportation de données de configuration, seuls les graphiques ayant été directement sélectionnés pour l'exportation sont exportés. Les graphiques pouvant être exportés sont enregistrés selon leur langue dans le portail TIA. Quand un projet est configuré en plusieurs langues, toutes les versions de langue utilisées sont exportées lors de l'exportation.

Lorsque vous exportez des graphiques, un nouveau dossier est créé dans le dossier du fichier d'exportation. Les graphiques exportés sont enregistrés dans ce dossier. Si ce dossier existe déjà, un nouveau dossier est créé et son nom est doté d'un numéro incrémenté.

Les graphiques sont enregistrés au même format de fichier que dans le projet. Le format n'est ni modifié, ni converti, et la résolution, ainsi que l'intensité des couleurs, restent également inchangées.

Pour la langue qui est choisie comme langue par défaut, c'est le code "default" qui est utilisé comme extension.

Quand un fichier de même nom se trouve déjà dans le dossier, un numéro incrémenté est ajouté au nom de fichier du graphique exporté.

Importation de graphiques

L'importation de graphiques implique les exigences suivantes :

- Les graphiques doivent avoir un format de fichier pris en charge par le portail TIA.
- Les graphiques doivent être référencés dans le fichier XML par une indication de chemin relative.

Lorsque vous exportez un graphique, vous pouvez l'éditer à l'aide d'une application graphique en dehors du portail TIA, puis le réimporter.

Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

8.2 Importation/exportation de données du projet

8.2.1 Exportation de textes de projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les textes de projet se trouvent sous le nœud "Langues et ressources" d'un projet dans TIA Portal. Cette fonction vous permet d'exporter des textes de projet dans un fichier xlsx, ce qui peut servir à des fins de traduction, par exemple. Les restrictions valables pour l'interface utilisateur s'appliquent également à l'exportation et l'importation de textes de projet.
Restrictions valables :

- Les textes exportés peuvent être importés uniquement dans le projet dont ils ont été exportés.
- Les textes ne peuvent être traduits que dans les langues existant dans le projet.
- Seuls les textes existants peuvent être réimportés. Une fois que des textes ont été supprimés du projet d'origine ou créés à nouveau, l'importation de ces textes échoue.

Vous devez définir les paramètres suivants :

Nom	Exemple	Description
fileName	"D:\Test\ProjectText.xlsx"	Chemin complet du fichier d'exportation
sourceLanguage	new CultureInfo("en-US")	Langue de référence de laquelle le texte doit être traduit
targetLanguage	new CultureInfo("de-DE")	Langue de référence dans laquelle le texte doit être traduit

Des paramètres de l'exemple utilisé, il résulte le code de programme suivant pour l'exportation de textes de projet :

```
project.ExportProjectTexts (@"D:\Test\ProjectText.xlsx", new CultureInfo("en-US"), new
CultureInfo("de-DE"));
```

8.2.2 Importation de textes de projet

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Cette fonction vous permet d'importer des textes de projet d'un fichier xlsx, ce qui peut servir à des fins de traduction, par exemple. Les restrictions valables pour l'interface utilisateur s'appliquent également à l'exportation et l'importation de textes de projet. Restrictions valables :

- Les textes exportés peuvent être importés uniquement dans le projet dont ils ont été exportés.
- Les textes traduits peuvent être importés uniquement dans les langues disponibles dans le projet dont ils ont été exportés.
- Seuls les textes existants peuvent être réimportés. Une fois que des textes ont été supprimés du projet d'origine ou créés à nouveau, l'importation de ces textes échoue.

Vous devez définir les paramètres suivants :

Nom	Exemple	Description
fileName	"D:\Test\ProjectText.xlsx"	Chemin complet du fichier d'importation
updateSourceLanguage	true	A "true", le texte de la langue de référence est actualisé à l'aide du fichier d'exportation. A "false", le texte de la langue de référence n'est pas actualisé.

Des paramètres de l'exemple utilisé, il résulte le code de programme suivant pour l'importation de textes de projet :

```
ProjectTextResult result = project.ImportProjectTexts(@"D:\Test\ProjectText.xlsx", true);
```

L'importation de textes de projet fournit en retour un objet qui affiche l'état de l'importation et indique le chemin où le journal d'importation est enregistré. Pour accéder à ces propriétés, vous pouvez utiliser les codes suivants :

```
ProjectTextResultState resultState = result.State;
string logFilePath = result.Path;
```

8.2.3 Graphiques

8.2.3.1 Exporter les graphiques d'un projet

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Vous avez le choix entre exporter un graphique individuel ou exporter tous les graphiques de la bibliothèque de graphiques d'un projet dans toutes les langues. Un fichier XML contenant toutes les entrées du graphique du projet concernées est créé lors de l'exportation et référencé avec les graphiques exportés. Les graphiques concernés sont stockés avec le fichier XML dans le même répertoire du système de fichiers.

Pour que les graphiques exportés ("*.jpg", "*.bmp", "*.png", "*.ico" etc.) puissent être modifiés, ces graphiques ne sont pas protégés en écriture.

Code du programme : Exporter un graphique

Pour exporter le graphique requis, utilisez le code de programme suivant :

```
//Exports all language variants of a single grafic
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
MultiLingualGraphic graphic = graphicsComposition.Find("graphicName");
graphic.Export(@"D:\ExportFolder\graphicName.xml", ExportOptions.WithDefaults);
```

Code du programme : Exporter tous les graphiques

Pour exporter tous les graphiques de la bibliothèque de graphiques, modifiez le code de programme suivant :

```
//Exports all graphics of a graphic library
Project project = …;
MultiLingualGraphicComposition graphicsComposition = project.Graphics;
foreach(MultiLingualGraphic graphic in graphicsComposition)
{
    graphic.Export(String.Format(@"D:\Graphics\{0}.xml", graphic.Name),
ExportOptions.WithDefaults);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.2.3.2 Importer des graphiques dans un projet

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Un fichier XML est stocké avec les différentes versions linguistiques d'un graphique dans un répertoire de votre système de fichiers.

Vous pouvez référencer tous les graphiques dans un chemin relatif de votre fichier XML.

Vous pouvez désormais importer toutes les versions linguistiques d'un graphique contenu dans le fichier XML dans la bibliothèque de graphiques.

Veuillez également tenir compte de ce qui suit Importation de données de configuration (Page 211).

Code du programme

Pour importer un ou plusieurs graphiques, modifiez le code de programme suivant :

```
//Import all language variants of a single graphic
Project project = …;
MultiLingualGraphicComposition graphicComposition = project.Graphics;
graphicComposition.Import(@"D:\Graphics\Graphic1.xml", ImportOptions.Override);
```

Voir aussi

Bibliothèques standard (Page 34)

8.3 Importation/exportation de données d'un appareil IHM

8.3.1 Cycles

8.3.1.1 Exportation de cycles

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface API prend en charge l'exportation de tous les cycles d'un appareil IHM connu vers un fichier XML. La génération du fichier d'export correspondant indique que l'export est terminé.

Code du programme

Pour exporter des cycles d'un appareil IHM vers un fichier XML, modifiez le code de programme suivant :

```
//Exports cycles from an HMI device
private static void ExportCyclesFromHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    foreach(Cycle cycle in cycles)
       {
        cycle.Export(String.Format(@"C:\OpennessSamples\ExportFiles\" + cycle.Name +
".xml"), ExportOptions.WithDefaults);
       }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.1.2 Importer des cycles

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Si vous utilisez `ImportOptions.None`, le numéro de la composition (Composition count) vous permet de détecter les cycles effectivement importés. Vous avez accès à ces cycles importés.

Remarque

Les cycles standard ont des propriétés qui ne peuvent être éditées dans l'interface utilisateur. Si vous indiquez dans le fichier d'importation que ces propriétés doivent être modifiées, l'importation déclenche une NonRecoverableException et ferme TIA Portal.

Code du programme

Pour importer un cycle ou plusieurs cycles dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports cycles to an HMI device
private static void ImportCyclesToHMITarget(HmiTarget hmitarget)
{
    CycleComposition cycles = hmitarget.Cycles;
    string dirPathImport = @"C:\OpennessSamples\Import\";
    string cycleImportFileName = "CycleImport.xml";
    string fullFilePath = dirPathImport + cycleImportFileName;

    cycles.Import(fullFilePath, ImportOptions.None);
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

8.3.2 Table des variables

8.3.2.1 Exporter des tables de variables IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Établissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

Un fichier XML est exporté par table de variables IHM. L'API prend en charge ce processus d'exportation. L'exportation de tables de variables est aussi disponible dans les sous-dossiers.

Code du programme : Exporter toutes les tables de variables IHM à partir d'un dossier indiqué

Pour exporter toutes les tables de variables IHM d'un dossier défini, modifiez le code de programme suivant :

```
//Exports all tag tables from a tag folder
private static void ExportAllTagTablesFromTagFolder(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    foreach (TagTable table in tables)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(fullFilePath, ExportOptions.WithDefaults);

    }
}
```

Code du programme : Exporter une table de variables IHM

Pour exporter une seule table de variables IHM, modifiez le code de programme suivant :

```
//Exports a tag table from an HMI device
private static void ExportTagTableFromHMITarget(HmiTarget hmitarget)
{
    string tableName = "Tag table XYZ";
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;
    TagTable table = tables.Find(tableName);

    if (table != null)
    {
        string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Code du programme : Exporter toutes les tables de variables IHM

Pour exporter toutes les tables de variables IHM, modifiez le code de programme suivant :

```
//Exports all tag tables from an HMI device
private static void ExportAllTagTablesFromHMITarget(HmiTarget hmitarget)
{
    TagSystemFolder sysFolder = hmitarget.TagFolder;

    //First export the tables in underlying user folder
    foreach (TagUserFolder userFolder in sysFolder.Folders)
    {
        ExportUserFolderDeep(userFolder);
    }

    //then, export all tables in the system folder
    ExportTablesInSystemFolder(sysFolder);
}

private static void ExportUserFolderDeep(TagUserFolder rootUserFolder)
{
        foreach (TagUserFolder userFolder in rootUserFolder.Folders)
        {
            ExportUserFolderDeep(userFolder);
        }
        ExportTablesInUserFolder(rootUserFolder);
}

private static void ExportTablesInUserFolder(TagUserFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(fullFilePath, ExportOptions.WithDefaults);
     }
 }

private static void ExportTablesInSystemFolder(TagSystemFolder folderToExport)
{
     TagTableComposition tables = folderToExport.TagTables;
     foreach (TagTable table in tables)
     {
         string fullFilePath = string.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
         table.Export(fullFilePath, ExportOptions.WithDefaults);
     }
 }
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.2.2 Importer une table de variables IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Établissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir projet (Page 90)

Code du programme

Pour importer la table de variables IHM d'un fichier XML dans un dossier personnalisé ou un dossier système, modifiez le code de programme suivant :

```
//Imports a single HMI tag table from a XML file
private static void ImportSingleHMITagTable(HmiTarget hmitarget)
{
    TagSystemFolder folder = hmitarget.TagFolder;
    TagTableComposition tables = folder.TagTables;

    string fullFilePath = @"C:\OpennessSamples\TagTables\myExportedTagTable.xml";
    tables.Import(fullFilePath, ImportOptions.Override);
}
```

Importation erronée de variables

Si vous utilisez les caractères spéciaux suivants dans des noms de variables ou de variables référencées, l'importation de variables échoue :

- . (Point)
- \ (Barre oblique inversée)

Solution 1 :

Vérifiez avant une exportation que les noms des variables à exporter ou des variables référencées ne contiennent aucun point ou barre oblique inversée.

Solution 2 :

Ajoutez dans le fichier d'importation des guillemets aux noms des variables ou de variables référencées.

Exemple

- Nom de variable avec caractère spécial :
  <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date \DB_SFX0908_HMI1.Actual_Date_Time.Hour</name>
- Nom de variable avec caractère spécial, entre guillemets :
  <name>"Siemens.Simatic.Hmi.Utah.Tag.HmiTag:41000_Options_Time_Date \DB_SFX0908_HMI1.Actual_Date_Time.Hour"</name>

Voir aussi

Bibliothèques standard (Page 34)

8.3.2.3 Exporter des variables individuelles d'une table de variables IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Établissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir projet (Page 90)

Utilisation

Les types d'objet de modèle d'objet suivants peuvent exister sous la forme d'éléments subordonnés d'une variable HMI et sont pris en compte à l'exportation :

MultilingualText	Pour commentaire, TagValue, DisplayName
TagArrayMemberTag	Pour éléments de tableau IHM
TagStructureMember	Pour éléments de structure IHM
Événement	Pour événements configurés
MultiplexEntry	Pour les entrées multiplex configurées de variables

Code du programme

Pour exporter une seule variable d'une table de variables IHM vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a selected tag from a tag table
private static void ExportSelectedTagFromTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable mytable = tagFolder.TagTables.Find("MyTagTable");

    TagComposition containingTags = mytable.Tags;
    Tag myTag = containingTags.Find("MyTag");

    if (myTag != null)
    {
       string fullFilePath = string.Format(@"C:\OpennessSamples\Tags\{0}.xml", myTag.Name);
        myTag.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.2.4 Importer des variables individuelles d'une table de variables IHM

Conditions

- L'application Openness est connectée au portail TIA.
  VoirÉtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

Les types d'objet de modèle d'objet suivants peuvent exister sous la forme d'éléments subordonnés d'une variable HMI et être pris en compte à l'importation :

MultilingualText	Pour commentaire, TagValue, DisplayName
TagArrayMemberTag	Pour éléments de tableau IHM
TagStructureMember	Pour éléments de structure IHM
Événement	Pour événements configurés
MultiplexEntry	Pour les entrées multiplex configurées de variables

Code du programme

Pour importer une variable IHM dans une table de variables IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports a tag into a tag table
private static void ImportTagIntoTagTable(HmiTarget hmitarget)
{
    TagSystemFolder tagFolder = hmitarget.TagFolder;
    TagTable myTable = tagFolder.DefaultTagTable;
    TagComposition tagComposition = myTable.Tags;

    string fullFilePath = @"C:\OpennessSamples\Tags\myExportedTag.xml";
    tagComposition.Import(fullFilePath, ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.2.5 Particularités de l'importation/exportation de variables IHM

Introduction

L'exportation/importation des variables IHM suivantes présentent des particularités :

- Variables IHM externes avec liaison intégrée
- Variables IHM avec le type de données "UDT"

Codes de programme similaires

Le code de programme pour les variables IHM susmentionnées est presque identique aux codes de programme suivants :

- Code du programme : Exportation de variables IHM (Page 225)
- Code du programme : Importation de variables IHM (Page 226)

Conditions requises

- L'application Openness est connectée àTIA Portal.
  Voir Etablissement d'une connexion à TIA Portal (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Particularités de l'importation/exportation d'une variable IHM externe avec liaison intégrée

Lors de l'exportation d'une variable IHM externe avec liaison IHM intégrée, seule la liaison des variables IHM à la variable API est enregistrée dans le fichier d'exportation, à la place des données de variables API.

Avant l'importation, assurez-vous que l'API, les variables API correspondantes et la liaison intégrée à l'API correspondant sont présents dans le projet. Si tel n'est pas le cas, il faut créer ces éléments avant de lancer l'importation. Lors de l'importation consécutive de la variable IHM externe, la liaison à la variable API est réactivée.

Les noms des variables IHM externes au-delà de toutes les tables de variables d'un projet doivent être univoques. Si vous n'indiquez pas la table de variables correspondant à la variable IHM lors de l'importation, l'importation est annulée.

Pour importer une variable IHM externe avec liaison intégrée, utilisez la structure XML suivante :

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    <AttributeList>
        <Name>MyIntegratedHmiTag_1</Name>
    </AttributeList>
    <LinkList>
        <AcquisitionCycle TargetID="@OpenLink">
            <Name>1 s</Name>
        </AcquisitionCycle>
        <Connection TargetID="@OpenLink">
            <Name>HMI_Connection_MP277_300400</Name>    <- Must exist in the project
        </Connection>
        <ControllerTag TargetID="@OpenLink">
            <Name>Datablock_1.DBElement1</Name>        <- Must exist in the project
        </ControllerTag>
    </LinkList>
</Hmi.Tag.Tag>
```

Particularités de l'importation/exportation d'une variable IHM du type de données "UDT"

Lors de l'exportation d'une variable IHM du type de données "UDT", le raccourci vers le type de données est exporté. Pour l'importation, seuls les types de données versionnés sont pris en charge.

Les types de données doivent être enregistrés dans la bibliothèque de projet. Les types de données de la bibliothèque globale ne sont pas pris en charge.

Les règles suivantes doivent être respectées pour l'importation :

- Les types de données référencés doivent figurer dans la bibliothèque de projet. L'importation est annulée si le type de données ne figurent pas dans la bibliothèque de projet.
- Le type de données référencé doit être versionné. L'attribution de versions est prise en charge à partir de TIA Portal V13 SP1.
  Si le type de données n'est pas versionné, une exception est déclenchée.

Remarque

Le premier type de données trouvé est utilisé pour la résolution de la référence lors de l'importation.

Tenez compte ici des points suivants : Le répertoire racine de la bibliothèque est d'abord parcouru, puis les sous-dossiers.

Pour importer une variable IHM du type de données "UDT", utilisez la structure XML suivante :

```
<Hmi.Tag.Tag ID="1" CompositionName="Tags">
    ...
    <LinkList>
        <DataType TargetID="@OpenLink">
            <Name>HmiUdt_1 V 1.0.0</Name>    <- Must exist in the project library
        </DataType>
        ...
    </LinkList>
    ...
</Hmi.Tag.Tag>
```

8.3.3 Scripts VB

8.3.3.1 Exporter des scripts VB

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Tous les dossiers personnalisés de niveau inférieur sont pris en compte au cours de l'exportation. Pour chaque script VB exporté est créé un fichier XML spécifique.

Code du programme : Exporter un script VB

Pour exporter un script VB sélectionné d'un appareil IHM vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a single vbscript of an HMI device
private static void ExportSingleVBScriptOfHMITarget(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    VBScript vbScript = vbScripts.Find("MyVBScript");

    string fullFilePath = String.Format(@"C:\OpennessSamples\Export\Scripts\{0}.xml",
vbScript.Name);
    vbScript.Export(fullFilePath, ExportOptions.None);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.3.2 Exporter des scripts VB à partir d'un dossier

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Pour chaque script VB exporté est créé un fichier XML spécifique.

Code du programme : exporter un script VB d'un dossier personnalisé

Pour exporter un script VB d'un dossier personnalisé vers un fichier XML, modifiez le code de programme suivant :

```
//Exports vbscripts of a selected vbscript system folder
private static void ExportVBScriptOfSelectedVBScriptSystemFolder(HmiTarget hmitarget)
{
    string dirPathExport = @"C:\Export";
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptUserFolderComposition vbUserFolders = vbScriptFolder.Folders;
    VBScriptUserFolder vbUserFolder = vbUserFolders.Find("MyVBUserFolder");
    VBScriptComposition vbScripts = vbUserFolder.VBScripts; foreach (VBScript script in
vbScripts)
    {
       string fullPathExport = String.Format(@"C:\OpennessSamples\Export\Scripts\{0}.xml",
script.Name);
        script.Export(fullPathExport, ExportOptions.None);
    }
}
```

Code du programme : Exporter tous les scripts VB à partir d'un dossier système

Pour exporter tous les scripts VB du dossier système, modifiez le code de programme suivant :

```
//Exports all vbscripts by using a foreach loop
private static void ExportAllVBScripts(HmiTarget hmitarget)
{
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (null != vbScripts)
    {
        foreach (VBScript script in vbScripts)
        {
            String fullFilePath = String.Format(@"C:\OpennessSamples\Scripts\{0}.xml",
script.Name);
            script.Export(fullFilePath, ExportOptions.None);
        }
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.3.3 Importer des scripts VB

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les importations groupées sont prises en charge : Sinon, vous pouvez aussi utiliser un code de programme avec une boucle Foreach (Exporter des scripts VB (Page 229)).

Code du programme

Pour importer un script VB dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports a single vbscript to an HMI device
private static void ImportSingleVBScriptToHMITarget(HmiTarget hmitarget)
{
    string dirPathImport = @"C:\OpennessSamples\Import";
    VBScriptSystemFolder vbScriptFolder = hmitarget.VBScriptFolder;
    VBScriptComposition vbScripts = vbScriptFolder.VBScripts;
    if (null != vbScripts)
    {
        string fullPath = dirPathImport + "ImportTest_VBScript.xml";
        vbScripts.Import(fullPath, ImportOptions.None);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.4 Listes de textes

8.3.4.1 Exporter des listes de textes à partir d'un appareil IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'exportation de listes de textes et de graphiques inclut toutes les entrées des listes. Les listes de textes et de graphiques peuvent être exportées séparément.

Les listes de textes d'un appareil IHM sont exportées. Pour chaque liste de textes exportée, un fichier XML spécifique est créé.

Code du programme

Pour exporter des listes de textes d'un appareil IHM, modifiez le code de programme suivant :

```
//Export TextLists
private static void ExportTextLists(HmiTarget hmitarget)
{
    TextListComposition text = hmitarget.TextLists;
    foreach (TextList textList in text)
    {
        String fullFilePath = String.Format(@"C:\OpennessSamples\TextGraphic\TextLists
\{0}.xml", textList.Name);
        textList.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.4.2 Importer une liste de texte dans un appareil IHM

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface API prend en charge l'importation d'une liste de textes dans un appareil IHM depuis un fichier XML.

Code du programme

Pour importer une liste de textes dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports a single TextList
private static void ImportSingleTextList(HmiTarget hmitarget)
{
    TextListComposition textListComposition = hmitarget.TextLists;
    IList<TextList> importedTextLists = textListComposition.Import( @"C:\OpennessSamples
\Import\textListImport.xml", ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.4.3 Formats XML avancés pour l'exportation/importation de listes de textes

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion à TIA Portal (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- Exportation standard de listes de textes
  Voir Exporter des listes de textes à partir d'un pupitre opérateur (Page 232)
- Importation standard de listes de textes
  Voir Importer des listes de textes dans un pupitre opérateur (Page 233)

Utilisation

Une liste de textes peut aussi contenir des textes formatés Cela concerne pour l'essentiel les formatages suivants :

- Formatage de texte
- Références aux autres objets dans le texte

Les formatages textuels purs dans une liste de textes à exporter conduisent à un format d'exportation XML étendu. Les références aux objets sont exprimés sous la forme d'Open Links. De même que les listes de textes à importer avec des textes formatés.

Les formats d'exportation XML étendus peuvent aussi nettement se complexifier. A titre d'exemple, d'autres liens que le seul nom de l'objet peuvent parfois exister dans la liste de

textes, p. ex. via un Open Link vers une variable API d'un autre appareil. Si tel est le cas, toutes les informations doivent être codées en une chaîne de caractères pour supprimer l'Open Link.

```
<?xml version="1.0" encoding="utf-8"?>
<Document>
  <!-- ... -->
  <MultilingualText ID="4" CompositionName="Text">
   <AttributeList>
      <TextItems>
        <Value lang="en-US">
          <body >
            <p>
              Mary<field ref="0" />
            </p>
          </body>
           <fieldinfos >
             <fieldinfo name="0" domaintype="HMICommonTagDisplayFormat" >
               <reference TargetID="@OpenLink">
                 <name>Siemens.Simatic.Hmi.Utah.Tag.HmiTag:Tag_1</name>
               </reference>
               <domaindata>
                 <format displaytype="Decimal" length="1" formatpattern="9" />
               </domaindata>
             </fieldinfo>
           </fieldinfos>
        </Value>
      </TextItems>
   </AttributeList>
  </MultilingualText>
  <!-- ... -->
</Document>
```

8.3.5 Listes de graphiques

8.3.5.1 Exporter les listes de graphiques

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'exportation de listes de textes et de graphiques inclut toutes les entrées des listes. Les listes de textes et de graphiques peuvent être exportées séparément.

Un fichier XML est créé par liste de graphiques. Les objets graphiques globaux contenus dans les listes de graphiques sont exportés sous la forme d'Open Links.

Code du programme

Pour exporter des listes de graphiques d'un pupitre opérateur, modifiez le code de programme suivant :

```
//Exports GraphicLists
private static void ExportGraphicLists(HmiTarget hmitarget)
{
    GraphicListComposition graphic = hmitarget.GraphicLists;
    foreach (GraphicList graphicList in graphic)
    {
        string fullFilePath = String.Format(@"C:\OpennessSamples\TextGraphic\GraphicLists
\{0}.xml", graphicList.Name);
        graphicList.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.5.2 Importer les listes de graphiques

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

L'interface API prend en charge l'importation d'une liste de graphiques dans un appareil IHM depuis un fichier XML.

Tous les objets graphiques référencés de la liste de graphiques sont inclus dans l'importation. Les références aux graphiques globaux ne sont pas incluses. Si les graphiques globaux référencés existent dans le projet cible, les références aux graphiques globaux sont rétablies lors de l'importation.

Code du programme

Pour importer une liste de graphiques dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports a single GraphicList
private static void ImportSingleGraphicList(HmiTarget hmitarget)
{
    GraphicListComposition graphicListComposition = hmitarget.GraphicLists;
    IList<GraphicList> importedGraphicLists = graphicListComposition.Import( @"C:
\OpennessSamples\Import\graphicListImport.xm", ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.6 Connexions

8.3.6.1 Exporter des connexions

Conditions

- L'application Openness est connectée au portail TIA.
  VoirÉtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

L'interface API prend en charge l'exportation de toutes les liaisons d'un appareil IHM vers un fichier XML.

Remarque

Exporter des connexions intégrées

L'exportation de connexions intégrées n'est pas prise en charge.

Pour chaque connexion exportée, un fichier XML spécifique est créé.

Code du programme

Pour exporter toutes les connexions d'un appareil IHM vers un fichier XML, modifiez le code de programme suivant :

```
//Exports communication connections from an HMI device
private static void ExportCommunicationConnectionsFromHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    foreach(Connection conn in connections)
    {
        String fullFilePath = String.Format(String.Format(@"C:\OpennessSamples\Export
\{0}.xml", conn.Name));
        conn.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.6.2 Importation de connexions

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

L'interface API prend en charge l'importation de toutes les liaisons d'un appareil IHM dans un appareil IHM depuis un fichier XML. Si vous souhaitez importer plusieurs liaisons de communication, importez à chaque fois le fichier XML pour la connexion correspondante.

Remarque

Si vous importez une liaison dans un projet dans lequel une liaison intégrée est déjà configurée, cette liaison n'est pas écrasée. L'importation est annulée et une Exception est déclenchée.

Code du programme

Pour importer une seule liaison d'un appareil IHM dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports Communication connections to an HMI device
private static void ImportCommunicationConnectionsToHMITarget(HmiTarget hmitarget)
{
    ConnectionComposition connections = hmitarget.Connections;
    IList<Connection> importedConnectionLists = connections.Import(@"C:\OpennessSamples
\Import\ConnectionImport.xml", ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7 Vues

8.3.7.1 Vue d'ensemble des objets graphiques pouvant être exportés

Utilisation

Vous pouvez importer ou exporter les vues suivantes par le biais de Public API :

Tableau 8-4 Vues prises en charge

Objet	WinCC V14
Vue	x
Vue globale	x
Modèle de vue	x
Fenêtre permanente	x
Vue contextuelle	–
Vue coulissante	–

Vous pouvez importer ou exporter les objets graphiques suivants par le biais de Public API :

Tableau 8-5 Objets graphiques pris en charge

Zone	Type d'objet	WinCC V14
Objets simples	Ligne	x
	Ligne polygonale	x
	Polygone	x
	Ellipse	x
	Segment d'ellipse	–
	Segment de cercle	–
	Arc d'ellipse	–
	Arc de cercle	–
	Cercle	x
	Rectangle	x
	Connecteur	–
	Champ de texte	x
	Affichage graphique	x
	Tuyau	–
	Double raccord en T	–
	Raccord en T	–
	Coude	–

Zone	Type d'objet	WinCC V14
Eléments	Champ d'E/S	x
	Champ d'E/S graphique	x
	Champ de texte éditable	–
	Champ de liste	–
	Zone de liste déroulante	–
	Bouton	x
	Bouton rond	–
	Bouton-poussoir lumineux	x
	Commutateur	x
	Champ d'E/S symbolique	x
	Champ date/heure	x
	Bargraphe	x
	Bibliothèque d'icônes	x
	Curseur	x
	Barre de défilement	–
	Case à cocher	–
	Bouton d'option	–
	Instrument à aiguille	x
	Horloge	x
	Vue de l'espace mémoire	–
	Touches de fonction (touches programmables)	x
	Groupes	x
	Instances de bloc d'affichage	–

Zone	Type d'objet	WinCC V14
Controls	Fenêtre de vues	–
	Vue des utilisateurs	x
	Travail d'impression/Diagnostic de script	–
	Affichage caméra	–
	Affichage PDF	–
	Vue de recette	–
	Vue des alarmes	–
	Indicateur d'alarme	–
	Fenêtre d'alarmes	–
	Vue de courbes f(x)	–
	Vue de courbes f(t)	–
	Vue tabellaire	–
	Table des valeurs	–
	HTML Browser	–
	Media Player	–
	Diagnostic de voie	–
	WLAN - Réception	–
	Zone - Nom	–
	Zone - Signal	–
	Nom de la plage d'action	–
	Nom de la plage d'action (RFID)	–
	Signal de la plage d'action	–
	Etat de chargement	–
	Molette	–
	Indicateur d'aide	–
	Vue Sm@rtClient	–
	Visualisation/forçage	–
	Vue de l'espace mémoire	–
	Affichage de section de programme NC	–
	Vue de diagnostic système	–
	Fenêtre de diagnostic système	–

Voir aussi

Notions élémentaires sur l'importation/exportation (Page 199)

8.3.7.2 Exporter toutes les vues d'un appareil IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Un autre code de programme est nécessaire pour exporter toutes les vues agrégées de tous les dossiers personnalisés d'un appareil IHM.

Code du programme : Exporter toutes les vues d'un dossier personnalisé

Pour exporter les vues d'un dossier de vues personnalisé d'un appareil IHM et le dossier de vues système, modifiez le code de programme suivant :

```
//Exports all screens of an HMI device
private static void ExportScreensFromFolder(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = …;
    ScreenComposition screens = folder.Screens;
    foreach(Screen screen in screens)
    {
        screen.Export(String.Format(@"D:\Screens\{0}.xml", screen.Name),
            ExportOptions.WithDefaults);
    }
}
```

Code du programme : exporter toutes les vues d'un appareil IHM quel que soit l'utilisateur

Pour exporter toutes les vues, modifiez le code de programme suivant :

```
public static void ExportScreens(string screenPath, HmiTarget target)
{
    string exportFilePath = @"C:\OpennessSamples\Screens";

    foreach(Screen screen in target.ScreenFolder.Screens)
    {
        screen.Export(System.IO.Path.Combine(exportFilePath, screen.Name + ".xml"),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder folder in target.ScreenFolder.Folders)
    {
       ExportScreenUserFolder(System.IO.Path.Combine(exportFilePath, folder.Name), folder);
    }
}

private static void ExportScreenUserFolder(string exportFilePath,ScreenUserFolder folder )
{
    foreach(Screen screen in folder.Screens)
    {
        screen.Export(System.IO.Path.Combine(exportFilePath, screen.Name + ".xml"),
ExportOptions.WithDefaults);
    }
    foreach(ScreenUserFolder subFolder in folder.Folders)
    {
        ExportScreenUserFolder(System.IO.Path.Combine(exportFilePath, subFolder.Name),
subFolder);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.3 Exporter une vue à partir d'un dossier de vues

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les données suivantes d'une vue sont exportées :

Vue	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Ouvrir des liens	Template
Compositions	• Layers • Animations Toutes les animations basées sur Runtime Advanced configurées sont exportées. • Events Tous les événement basés sur Runtime Advanced configurés sont exportés. • Softkeys Toutes les touches programmables configurées sont exportées.

Pour chaque couche, les données suivantes sont exportées :

Remarque

le nom de la couche dans TIA Portal est un texte vide par défaut.

Si vous ne modifiez pas le nom de la couche dans TIA Portal, le nom de la couche exportée est vide. Dans ce cas, le nom de la couche affiché dans TIA Portal dépend de la langue de l'interface utilisateur.

Si vous modifiez le nom de la couche dans TIA Portal, le nom modifié sera affiché dans toutes les langues correspondantes.

Couche	Données
Propriétés	Name, Index, VisibleES
Compositions	ScreenItems (avec éléments de vue)

Les éléments suivants ne sont pas inclus dans l'exportation :

- Propriétés spécifiques à SCADA
- Couches qui ne contiennent pas d'éléments de vue et dont les propriétés ne peuvent se distinguent des valeurs par défaut.

Code du programme

Pour exporter une seule vue à partir du dossier utilisateur ou du dossier système d'un appareil IHM, modifiez le code de programme suivant :

```
//Exports a single screen from a screen folder
private static void ExportSingleScreenFromScreenFolder(HmiTarget hmitarget)
{
    string screenName = "Screen_1.xml";
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    //or ScreenSystemFolder folder = hmitarget.ScreenFolder;
    ScreenComposition screens = folder.Screens;
    Screen screen = screens.Find(screenName);
    if (screen != null)
    {
        screen.Export(String.Format(@"C:\OpennessSamples\Screens\{0}.xml", screen.Name),
            ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.4 Importer des vues dans un appareil IHM

Conditions

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les vues ne peuvent être importées que dans un type donné d'appareil IHM. L'appareil IHM et l'appareil à partir duquel les vues ont été exportées sont du même type d'appareil.

Les données suivantes d'une vue sont exportées :

Vue	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name, Number, HelpText
Ouvrir des liens	Templates
Compositions	• Layers • Animations Toutes les animations configurables pour des vues sont importées. • Events Toutes les animations configurables pour des événements sont importées. • Softkeys Toutes les animations configurables pour des touches programmables sont importées.

Pour chaque couche, les données suivantes sont importées :

Remarque

Si vous avez indiqué un texte vide pour le nom de la couche avant l'importation, le nom de la couche affiché dans TIA Portal dépend de la langue de l'interface utilisateur après l'importation.

Si vous avez attribué un nom à la couche, le nom indiqué est affiché dans toutes les langues correspondantes après l'importation.

Couche	Données
Propriétés	Name, Index
Compositions	ScreenItems

Restrictions :

- Lorsque la largeur et la hauteur d'une vue ne correspondent pas aux dimensions de l'appareil, le processus d'importation est interrompu et une Exception est déclenchée. L'ajustement des éléments de vue compris n'est pas pris en charge. C'est pourquoi, certains éléments de vue peuvent se trouver en-dehors des limites de la vue. Si tel est le cas, un avertissement du compilateur est émis.
- Le numéro de vue doit être univoque pour toutes les vues de l'appareil. L'importation d'une vue est annulée si une vue avec un numéro de vue qui a déjà été créé dans l'appareil, est trouvé. Si vous n'avez pas encore attribué de numéro à la vue, un numéro de vue univoque est affecté à la vue pendant le processus d'importation.
- L'ordre des éléments de vue au sein de l'ordre Z doit être univoque et sans lacunes pour chaque couche dans la vue. C'est pourquoi une vérification de la cohérence, qui répare l'ordre si nécessaire, est effectuée après l'importation de la vue. Ce processus peut entraîner la modification d'"Indices de tabulation" pour certains éléments de vue.
  Vous pouvez modifier manuellement l'ordre Z des éléments de vue dans le fichier XML. L'élément de vue au premier emplacement se trouve tout à la fin de l'ordre Z.

Remarque

Vous pouvez modifier les valeurs de largeur et hauteur d'un élément de vue dans le fichier XML si la propriété "Adapter la taille au contenu" est activée pour l'élément de vue.

Remarque

L'importation de types de vue de la bibliothèque n'est pas prise en charge

À partir de WinCC V12 SP1, vous pouvez créer une vue en tant que type dans la bibliothèque. Les instances du type de vue utilisées dans le projet peuvent être éditées avec l'application Openness comme d'autres vues. Si vous exportez des vues, les instances des types de vue sont exportées sans les informations de type.

Si vous réimportez ces vues dans le projet, les instances du type de vue sont écrasées et l'instance est résolue par le type de vue.

Code du programme : Importer des vues dans un appareil IHM

Pour importer des vues avec la boucle For each dans un appareil IHM, modifiez le code de programme suivant :

```
//Imports all screens to an HMI device
private static void ImportScreensToHMITarget(HmiTarget hmitarget)
{
    string[] exportedScreens = new string[]{"Screen_1.xml", "Screen_n.xml"};
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Find("MyScreenFolder");
    foreach (string screenFileName in exportedScreens)
    {
        folder.Screens.Import(String.Format(@"D:\Screens\{0}.xml"), screenFileName),
ImportOptions.Override);
    }
}
```

Code du programme : Importer dans un dossier utilisateurs nouvellement créé

Pour importer une vue dans un dossier utilisateur venant d'être créé d'un appareil IHM, modifiez le code de programme suivant :

```
//Imports a single screen to a new created user folder of an HMI device
private static void ImportSingleScreenToNewFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenUserFolder folder = hmitarget.ScreenFolder.Folders.Create("MyFolder");
    folder.Screens.Import(@"C:\OpennessSamples\Import\ExportedScreens.xml",
ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.5 Exporter une fenêtre permanente

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les données suivantes de la fenêtre permanente sont exportées :

Fenêtre permanente	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name
Compositions	Layers

Pour chaque couche, les données suivantes sont exportées :

Couche	Données
Propriétés	Name, Index
Compositions	ScreenItems (avec éléments de vue)

Code du programme

Pour exporter une fenêtre permanente d'un appareil IHM vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a screenoverview
private static void ExportScreenoverview(HmiTarget hmitarget)
{
    ScreenOverview overview = hmitarget.ScreenOverview;
    if (overview != null)
    {
        overview.Export(@"C:\OpennessSamples\ExportedOverview.xml",
ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.6 Importer une fenêtre permanente

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les données suivantes de la fenêtre permanente sont importées :

Fenêtre permanente	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name, Visible, Number
Compositions	Layers

Pour chaque couche, les données suivantes sont importées :

Couche	Données
Propriétés	Name, Index
Compositions	ScreenItems (avec éléments de vue)

Lorsque la largeur et la hauteur d'une vue ne correspondent pas aux dimensions de l'appareil, le processus d'importation est interrompu et une Exception est déclenchée. L'ajustement des éléments d'appareil compris (éléments de vue) n'est pas pris en charge. C'est pourquoi, certains éléments d'appareil peuvent se trouver en-dehors des limites de la vue. Si tel est le cas, un avertissement du compilateur est émis.

L'ordre des éléments d'appareil dans la fenêtre permanente doit être univoque et ne présenter aucune lacune. C'est pourquoi une vérification de la cohérence, qui répare l'ordre si nécessaire, est effectuée après l'importation de la fenêtre permanente. Ce processus peut entraîner la modification d'"Indices de tabulation" pour certains éléments d'appareil.

Code du programme

Pour importer une fenêtre permanente dans un appareil IHM depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports a screenoverview
private static void ImportScreenOverview(HmiTarget hmiTarget)
{
    hmiTarget.ImportScreenOverview(@"C:\OpennessSamples\ExportedOverview.xml",
ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.7 Exporter des modèles de vue à partir d'un dossier

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les données suivantes du modèle de vue sont exportées :

Modèles de vue	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name
Compositions	• Layers • Animations Toutes les animations configurées sont exportées. Les animations SCADA ne sont pas exportées. • Softkeys Toutes les touches programmables configurées sont exportées.

Pour chaque couche, les données suivantes sont exportées :

Couche	Données
Propriétés	Name, Index
Compositions	ScreenItems (avec éléments de vue)

Code du programme : Exporter un modèle de vue

Pour exporter un seul modèle de vue à partir du dossier système ou d'un dossier personnalisé, modifiez le code de programme suivant :

```
//Exports a single screen template
private static void ExportSingleScreenTemplate(HmiTarget hmiTarget)
{
    string templateName = "Template name XYZ";
    string fullFilePath = String.Format(@"C:\OpennessSamples\Templates\{0}.xml",
templateName);

    ScreenTemplateUserFolder folder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    //or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    ScreenTemplate template = templates.Find(templateName);
    if(template != null)
    {
        template.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Code du programme : exporter tous les modèles de vue

Pour exporter tous les modèles de vue d'un certain dossier, modifiez le code de programme suivant :

```
//Exports all screen templates of a selected folder
private static void ExportAllScreenTamplates(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder folder =
hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    //or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    ScreenTemplateComposition templates = folder.ScreenTemplates;
    foreach(ScreenTemplate template in templates)
    {
        string fullFilePath = String.Format(@"C:\OpennessSamples\Templates\{0}.xml",
template.Name);
        template.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.8 Exporter tous les modèles de vue d'un appareil IHM

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Un fichier XML est créé par modèle de vue.

Les exportations groupées n'étant pas prises en charge, vous devez énumérer tous les modèles de vue et les exporter séparément. Ce faisant, veillez à ce que les noms utilisés pour les modèles de vue correspondent aux conventions de dénomination de fichiers de votre système de fichiers.

Code du programme

Pour exporter un modèle de vue à partir d'un appareil IHM, modifiez le code de programme suivant :

```
public static void ExportScreenTemplates(HmiTarget hmiTarget)
{
    string templatePath = @"C:\OpennessSamples\Templates";

    foreach (ScreenTemplate screen in target.ScreenTemplateFolder.ScreenTemplates)
    {
        screen.Export(Path.Combine(templatePath, screen.Name + ".xml"),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder folder in target.ScreenTemplateFolder.Folders)
    {
        ExportTemplateUserFolder(Path.Combine(templatePath, folder.Name), folder);
    }
}

private static void ExportTemplateUserFolder(string exportFilePath,
ScreenTemplateUserFolder folder)
{
    foreach (ScreenTemplate screen in folder.ScreenTemplates)
    {
        screen.Export(Path.Combine(exportFilePath, screen.Name + ".xml"),
ExportOptions.WithDefaults);
    }
    foreach (ScreenTemplateUserFolder subFolder in folder.Folders)
    {
        ExportTemplateUserFolder(Path.Combine(exportFilePath, subFolder.Name + ".xml"),
subFolder);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.3.7.9 Importer des modèles de vue

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)

Utilisation

Les données suivantes d'un modèle de vue sont importées :

Modèle de vue	Données
Propriétés	ActiveLayer, BackColor, Height, Width, Name, SetTabOrderInFront
Compositions	• Layers • Animations Toutes les animations configurables pour des vues sont importées. • Softkeys Toutes les animations configurables pour des touches programmables sont importées.

Pour chaque couche, les données suivantes sont importées :

Couche	Données
Propriétés	Name, Index
Compositions	ScreenItems (avec éléments de vue)

Lorsque la largeur et la hauteur d'un modèle de vue ne correspondent pas aux dimensions de l'appareil, le processus d'importation est interrompu et une Exception est déclenchée. L'ajustement des éléments de vue compris n'est pas pris en charge. C'est pourquoi, certains éléments de vue peuvent se trouver en-dehors des limites de la vue. Si tel est le cas, un avertissement du compilateur est émis.

L'ordre des éléments d'appareil dans le modèle de vue doit être univoque et ne présenter aucune lacune. C'est pourquoi une vérification de la cohérence, qui répare l'ordre si nécessaire, est effectuée après l'importation du modèle de vue. Ce processus peut entraîner la modification d'"Indices de tabulation" pour certains éléments de vue.

Code du programme : Importation générale

Pour importer tous les modèles de vues avec la boucle For each dans un appareil IHM, modifiez le code de programme suivant :

```
//Imports screen templates to an HMI device
private static void ImportScreenTemplatesToHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder folder = …;
    hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    // or ScreenTemplateSystemFolder folder = hmitarget.ScreenTemplateFolder;
    string[] exportedTemplates = {"Template_1.xml", "Template_2.xml", "Template_3.xml"};
    foreach (string templateFileName in exportedTemplates)
    {
        folder.ScreenTemplates.Import(String.Format(@"D:\Templates
\{0}.xml",templateFileName), ImportOptions.Override);
    }
}
```

Code du programme : Importer dans un dossier utilisateurs nouvellement créé

Pour importer un modèle de vue dans un dossier utilisateur venant d'être créé d'un appareil IHM, modifiez le code de programme suivant :

```
//Imports screen templates to a user folder of an HMI device
private static void ImportScreenTemplatesToFolderOfHMITarget(HmiTarget hmitarget)
{
    ScreenTemplateUserFolder screenTemplateFolder =
    hmitarget.ScreenTemplateFolder.Folders.Find("MyTemplateFolder");
    ScreenTemplateUserFolder folder = screenTemplateFolder.Folders.Create("MyNewFolder");
    folder.ScreenTemplates.Import(@"C:\ScreenTemplates\ExportedScreenTemplate.xm",
ImportOptions.Override);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.4 Importation/exportation de données d'un appareil API

8.4.1 Blocs

8.4.1.1 Modifications du modèle d'objet et format de fichier XML

Introduction

Pour importer un fichier XML personnalisé créé par l'utilisateur ou édité par le bias d'Openness avec succès dans TIA Portal, le fichier doit respecter des schémas définis.

Chaque fichier XML se compose de deux parties principales :

- Interface
- Unité de compilation

La section qui suit propose une description des schémas à respecter par les fichiers.

Interface

Une interface peut contenir plusieurs segments (par ex. Input, InOut, Static) : Vous trouverez tous ces segments dans le répertoire suivant :

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas \SW.InterfaceSections.xsd

Unité de compilation

Il existe des schémas différents pour les unités de compilation des blocs GRAPH, LAD/FBD et STL. Vous trouverez les schémas correspondants dans les répertoires suivants :

- GRAPH : C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas \SW.PlcBlocks.Graph.xsd
- LAD/FBD : C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas \SW.PlcBlocks.LADFBD.xsd
- STL : C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas \SW.PlcBlocks.STL.xsd

Sous-schémas

Il existe les définitions de schéma supplémentaires suivantes, utilisées par toutes les unités de compilation :

- Accès
- Généralités

Accès

Le nœud d'accès décrit par ex. :

- membres locaux/globaux et utilisations constantes
- Appels de FB, FC, d'instructions
- DB pour les appels

Vous trouverez le schéma d'accès dans le répertoire suivant :

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas \SW.PlcBlocks.Access.xsd

Généralités

Regroupe les attributs et éléments généraux utilisés, par ex. commentaires de toutes sortes, textes et jetons.

Vous trouverez le schéma général dans le répertoire suivant :

C:\Program Files\Siemens\Automation\Portal V14\PublicAPI\Schemas\SW.Common.xsd

8.4.1.2 Exporter des blocs

Conditions

- L'application Openness est connectée à TIA Portal.
  Voir Établissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

L'interface API prend en charge l'exportation de blocs et types de données utilisateur cohérents vers un fichier XML.

Le fichier XML se voit attribuer le nom du bloc. Les types de bloc suivants sont pris en charge :

- Blocs fonctionnels (FB)
- Fonctions (FC)
- Blocs d'organisation (OB)
- Blocs de données globaux (DB)

Les types de langages de programmation suivants sont pris en charge :

- LIST
- LOG
- CONT

- GRAPH
- SCL
  Le fichier XML issu de l'exportation d'un bloc SCL contient uniquement l'interface de bloc.

Données exportées

Les propriétés suivantes peuvent être exportées selon le bloc ou le type de données utilisateur et selon les `ExportOptions` sélectionnées (voir Exportation de données de configuration (Page 210)). Les propriétés représentées en gras sont toujours exportées.

Propriété	Type	Valeur par défaut	Protection en écriture	Domaine de validité
Name	String	numéro suivant disponible	false	Tous les blocs
Number	Int32	-	false	Tous les blocs
AutoNumber	Bool	true	false	Tous les blocs
HeaderAuthor	String	""	false	Tous les blocs
HeaderFamily	String	""	false	Tous les blocs
HeaderName	String	""	false	Tous les blocs
HeaderVersion	String	"0,1"	false	Tous les blocs
MemoryLayout	enum MemoryLayout	-	false	Tous les blocs
IsWriteProtectedInAS	Bool	false	false	Seulement DB
IsOnlyStoredInLoadMemory	Bool	false	false	Seulement DB
IsIECCheckEnabled	Bool	false	false	Seulement DB et FB
IsRetainMemResEnabled[1]	Bool	false	false	Seulement DB et FB
MemoryReserve	Unsigned	0	false	Seulement DB et FB
RetainMemoryReserve[1]	Unsigned	0	false	Seulement DB et FB
SecondaryType[2]	String	-	false	Seulement OB
ProgrammingLanguage	enum ProgrammingLanguage	-	false	Tous les blocs
IsKnowHowProtected	Bool	false	true	Tous les blocs
InstanceOfName	String	""	false	Seulement le DB d'instance d'un FB et le DB d'instance d'un UDT
InstanceOfNumber	Unsigned Short	-	true	Seulement le DB d'instance d'un FB et le DB d'instance d'un UDT
InstanceOfType	enum BlockType	-	true	Seulement le DB d'instance d'un FB et le DB d'instance d'un UDT
OfSystemLibElement	String	""	false	Seulement le DB d'instance d'un FB et le DB d'instance d'un UDT

Propriété	Type	Valeur par défaut	Protection en écriture	Domaine de validité
OfSystemLibVersion	String	""	false	Seulement le DB d'instance d'un FB et le DB d'instance d'un UDT
ParameterPassing	Bool	false	false	Seulement FB et FC (LIST)
CreationDate	DateTime	-	true	Tous les blocs
ModifiedDate	DateTime	-	true	Tous les blocs
Interface	String	interface non occupée	false	Tous les blocs
InterfaceModifiedDate	DateTime	-	true	Tous les blocs
ParameterModified	DateTime	-	true	Tous les blocs
StructureModified	DateTime	-	true	Tous les blocs
CodeModifiedDate	DateTime	-	true	Tous les blocs
CompileDate	DateTime	-	true	Tous les blocs
IsConsistent	Bool	-	true	Tous les blocs
UDAEnableTagReadback	Bool	false	false	Seulement le FB, la FC et le DB d'instance d'un FB
UDABlockProperties	String	""	false	Seulement le FB, la FC et le DB d'instance d'un FB
HandleErrorsWithinBlock	Bool	false	true	Seulement FB et OB
PLCSimAdvancedSupport	Bool	false	true	Tous les blocs
LibraryConformanceStatus	String	-	false	Seulement FB et FC
IsPLCDB	Bool	false	false	Seulement DB
SkipSteps	Bool	false	false	Seulement blocs GRAPH
AcknowledgeErrorsRequired	Bool	true	false	Seulement blocs GRAPH
PermanentILProcessingInMAN-Mode	Bool	false	false	Seulement blocs GRAPH
LockOperatingMode	Bool	false	false	Seulement blocs GRAPH
SetENOAutomatically	Bool	-	false	Seulement blocs GRAPH
CreateMinimizedDB	Bool	false	false	Seulement blocs GRAPH
LanguageInNetworks	String	-	false	Seulement blocs GRAPH
ISMultiInstanceCapable	Bool	-	true	Seulement FB
ArrayDataType	String	-	true	Seulement DB
ArrayLimitUpperBound	Int32	-	true	Seulement DB
GraphVersion	String	-	false	Tous les blocs
WithAlarmHandling	Bool	true	false	Tous les blocs

Propriété	Type	Valeur par défaut	Protection en écriture	Domaine de validité
DownloadWithoutReinit	Bool	-	false	Pas de GRAPH s'il ne s'agit pas d'un produit de sécurité
LibraryType	String	-	true	Seulement FB et FC
LibraryTypeVersionGuid	String	-	true	Seulement FB et FC
ILibraryTypeInstance.Connected-Version	ILibraryTypeVersion	-	false	Seulement FB et FC
ILibraryTypeInstance.Dependencies	ILibraryTypeInstanceAssociation	-	false	Seulement FB et FC
ILibraryTypeInstance.Dependents	ILibraryTypeInstanceAssociation	-	false	Seulement FB et FC

[1] Si la propriété "IsRetainMemResEnabled possède la valeur false et que la propriété "RetainMemoryReserve" n'est pas égale à 0, le système déclenche une exception.

[2] Lors de l'exportation d'un OB, le SecondaryType est défini par ailleurs à l'aide du numéro d'OB. L'affectation est vérifiée pendant le processus d'importation. Si l'affectation est incorrecte, une exception de type "Recoverable" est déclenchée.

Vous trouverez des informations complémentaires dans le système d'informations de TIA Portal sous "Aperçu des propriétés du bloc".

Code du programme

Pour exporter un bloc dépourvu de protection Know-how vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a regular block
private static void ExportRegularBlock(PlcSoftware plcSoftware)
{
    PlcBlock plcBlock = plcSoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(String.Format(@"C:\OpennessSamples\Blocks\{0}.xml", plcBlock.Name),
ExportOptions.WithDefaults);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.4.1.3 Exporter des blocs avec protection know-how

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

Le fichier XML qui en résulte ressemble au fichier d'exportation d'un bloc sans protection know-how. L'exportation couvre toutefois les données visibles de l'interface utilisateur lorsque le bloc s'ouvre sans mot de passe.

La liste d'attributs du bloc indique que le bloc correspondant possède une protection know-how.

Code du programme

Pour exporter les données visibles d'un bloc doté d'une protection know-how vers un fichier XML, modifiez le code de programme suivant :

```
private static void ExportBlock(PlcSoftware plcsoftware)
{
    PlcBlock plcBlock = plcsoftware.BlockGroup.Blocks.Find("MyBlock");
    plcBlock.Export(String.Format(@"C:\OpennessSamples\Blocks\{0}.xml", plcBlock.Name),
ExportOptions.WithDefaults);
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.4.1.4 Exporter des blocs F

Remarque

Les blocs F sont exportés de la même manière que les autres blocs.

Voir aussi

Etablissement d'une connexion au portail TIA (Page 69)

Ouvrir un projet (Page 90)

Bibliothèques standard (Page 34)

Exporter des blocs (Page 257)

Exporter des blocs avec protection know-how (Page 260)

8.4.1.5 Exporter des blocs système

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)
- Le projet contient un bloc système.
- L'API n'est pas en ligne.

Utilisation

La liste d'attributs du bloc indique que le bloc correspondant possède une protection know-how.

Code du programme

Pour exporter les données visibles d'un bloc vers un fichier XML, modifiez le code de programme suivant :

```
//Exports system blocks
private static void ExportSystemBlocks(PlcSoftware plcsoftware)
{
    PlcSystemBlockGroup sbSystemGroup = plcsoftware.BlockGroup.SystemBlockGroups[0];
    foreach (PlcSystemBlockGroup group in sbSystemGroup.Groups)
    {
        foreach (PlcBlock block in group.Blocks)
        {
            String fullFilePath = String.Format(@"C:\OpennessSamples\Blocks\{0}.xml",
block.Name);
            block.Export(fullFilePath, ExportOptions.WithDefaults);
        }
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

8.4.1.6 Importer un bloc

Conditions requises

- Pour importer des blocs, vous avez besoin des fichiers suivants :
  - Fichier de validation XML
  - Fichier d'utilisation XML

  Vous trouverez des informations détaillées sur la manière d'obtenir et d'utiliser ces fichiers à l'endroit suivant : Enabler File et Usage File (Page 25)
- L'application Openness est liée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

La Public API prend en charge l'importation de blocs avec les langages de programmation "LIST", "LOG", "GRAPH" ou "CONT" à partir d'un fichier XML. Les types de bloc suivants sont pris en charge :

- Blocs fonctionnels (FB)
- Fonctions (FC)
- Blocs d'organisation (OB)
- Blocs de données globaux (DB)

Remarque

S'il n'existe aucun "BlockNumber", le numéro de bloc est automatiquement affecté.

S'il n'existe aucune "Version", le numéro de version actuel est utilisé.

Remarque

Importation de blocs de données optimisés

Les blocs de données optimisés sont uniquement pris en charge par les CPU à partir de S7-1200. Si vous importez des blocs de données optimisés dans une S7-300 ou S7-400, ces blocs de données seront signalés comme "incohérents".

Remarque

L'utilisation de constantes globales peut être importée même si la constante globale n'est pas définie.

Réaction à l'importation

Pour l'importation d'un bloc, les règles à respecter sont les suivantes :

- Le fichier XML peut contenir moins de données que le bloc se trouvant dans le projet, par ex. moins de paramètres.
- Les informations redondantes telles que les informations d'appel doivent être identiques dans le projet et dans le fichier XML. Faute de quoi une exception est déclenchée.
- Les données du fichier XML peuvent être "incohérentes" en ce qui concerne leur capacité à être compilées dans TIA Portal.
- Les attributs possédant les propriétés "ReadOnly=True" et "Informative=True" ne sont pas importés.
- Les DB d'instance manquants ne sont pas automatiquement créés.

Code du programme

Pour importer un bloc à partir d'un fichier XML, modifiez le code de programme suivant :

```
//Imports blocks
private static void ImportBlocks(PlcSoftware plcSoftware)
{
    string filePath = @"C:\OpennessSamples\Import\ SystemBlocks.xml”;
    PlcBlockSystemGroup blockGroup = plcSoftware.BlockGroup;
    IList<PlcBlock> blocks = blockGroup.Blocks.Import(filePath, ImportOptions.Override);
}
```

8.4.2 Tables des variables

8.4.2.1 Exporter des tables de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

Un fichier XML est exporté par table des variables API.

L'interface Public API prend en charge l'exportation de toutes les tables de variables API du groupe système et de ses sous-groupes.

Code du programme

Pour exporter toutes les tables de variables API du groupe système et de ses sous-groupes, modifiez le code de programme suivant :

```
private static void ExportAllTagTables(PlcSoftware plcSoftware)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    // Export all tables in the system group
    ExportTagTables(plcTagTableSystemGroup.TagTables);
    // Export the tables in underlying user groups
    foreach(PlcTagTableUserGroup userGroup in plcTagTableSystemGroup.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}

private static void ExportTagTables(PlcTagTableComposition tagTables)
{
    foreach(PlcTagTable table in tagTables)
    {
        string fullFilePath = String.Format(@"C:\OpennessSamples\TagTables\{0}.xml",
table.Name);
        table.Export(fullFilePath, ExportOptions.WithDefaults);
    }
}

private static void ExportUserGroupDeep(PlcTagTableUserGroup group)
{
    ExportTagTables(group.TagTables);
    foreach(PlcTagTableUserGroup userGroup in group.Groups)
    {
        ExportUserGroupDeep(userGroup);
    }
}
```

Voir aussi

Bibliothèques standard (Page 34)

Exportation de données de configuration (Page 210)

8.4.2.2 Importer une table de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Code du programme

Pour importer des tables de variables API ou une structure de dossiers avec des tables de variables API dans le groupe système ou un groupe personnalisé depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports tag tables to the tag system group
private static void ImportTagTable(PlcSoftware plcSoftware)
{
    string exportFileName = "ExportedTagTable.xml";
    string dirPathImport = @"C:\OpennessSamples\Import";
    string fullFilePath = Path.Combine(dirPathImport, exportFileName);
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTagTableComposition tagTables = plcTagTableSystemGroup.TagTables;
    tagTables.Import(fullFilePath, ImportOptions.Override);
    // Or, to import into a subfolder:
    // plcTagTableSystemGroup.Groups.Find("SubGroup").TagTables.Import(fullFilePath,
ImportOptions.Override);
}
```

Voir aussi

Remarques sur la performance de TIA Portal Openness V14 (Page 35)

Bibliothèques standard (Page 34)

8.4.2.3 Exporter des variables ou constantes individuelles d'une table de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  VoirEtablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir projet (Page 90)

Utilisation

L'interface API prend en charge l'exportation d'une variable ou d'une constante depuis une table de variables API vers un fichier XML. Ce faisant, veillez à ce que les noms utilisés pour les tables des variables correspondent aux conventions de dénomination de fichiers de votre système de fichiers.

Le commentaire d'une variable ou d'une constante n'est exporté que si au moins une langue est définie pour le commentaire. Si le commentaire n'est pas défini pour toutes les langues du projet, il est uniquement exporté pour les langues du projet définies.

Code du programme

Pour exporter une variable ou une constante déterminée d'une table de variables API vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a single tag or constant of a controller tag table
private static void ExportTag(PlcSoftware plcSoftware, string tagName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcTag tag = plcTagTableSystemGroup.TagTables[0].Tags.Find(tagName);
    if(tag != null)
    {
        tag.Export(String.Format(@"C:\OpennessSamples\SingleTags\{0}.xml", tag.Name),
ExportOptions.WithDefaults);
    }
}

private static void ExportUserConstant(PlcSoftware plcSoftware, string userConstantName)
{
    PlcTagTableSystemGroup plcTagTableSystemGroup = plcSoftware.TagTableGroup;
    PlcConstant plcConstant =
plcTagTableSystemGroup.TagTables[0].Constants.Find(userConstantName); if(plcConstant!=
null)
    {
      plcConstant.Export(String.Format(@"C:\OpennessSamples\SingleUserConstants\{0}.xml",
plcConstant.Name), ExportOptions.WithDefaults);
    }
}
```

Voir aussi

Exportation de données de configuration (Page 210)

Remarques sur la performance de TIA Portal Openness V14 (Page 35)

Bibliothèques standard (Page 34)

8.4.2.4 Importer une seule variable ou constante dans une table de variables API

Conditions requises

- L'application Openness est connectée au portail TIA.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouverture d'un projet (Page 90)

Utilisation

Lors d'un appel d'importation, vous pouvez importer soit des variables soit des constantes.

Code du programme

Pour importer des groupes de variables ou certaines variables ou constantes depuis un fichier XML, modifiez le code de programme suivant :

```
//Imports tags into a plc tag table
private static void ImportTag(PlcSoftware plcSoftware, string tagtableName, string tagFile)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware.TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable != null)
    {
        tagTable.Tags.Import(tagFile, ImportOptions.Override);
    }
}

//Imports constants into a plc tag table
private static void ImportConstant(PlcSoftware plcSoftware, string tagtableName,
stringconstantFile)
{
    PlcTagTableSystemGroup plcTagTableSystemgroup = plcSoftware. TagTableGroup;
    PlcTagTable tagTable = plcTagTableSystemgroup.TagTables.Find(tagtableName);
    if(tagTable != null)
    {
        tagTable.Constants.Import(constantFile, ImportOptions.Override);
    }
}
```

Voir aussi

Exportation de données de configuration (Page 210)

Remarques sur la performance de TIA Portal Openness V14 (Page 35)

Bibliothèques standard (Page 34)

8.4.3 Exporter un type de données utilisateur

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Code de programme

Pour exporter un type de données utilisateur vers un fichier XML, modifiez le code de programme suivant :

```
//Exports a user defined type
private static void ExportUserDefinedType(PlcSoftware plcSoftware)
{
    string udtname = "udt name XYZ";
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    PlcType udt = types.Find(udtname);
    udt.Export(String.Format(@"C:\OpennessSamples\udts\{0}.xml", udt.Name),
ExportOptions.WithDefaults);
}
```

8.4.4 Importer un type de données utilisateur

Conditions requises

- L'application Openness est connectée au portail TIA.
  Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

L'interface API prend en charge l'importation de types de données utilisateur à partir d'un fichier XML.

Syntaxe du fichier d'importation

L'exemple de code suivant présente un extrait d'un fichier d'importation d'un type de données utilisateur :

```
<Section Name="Input">
 <Member Name="Input1" Datatype=quot;myudt1&quot;>
  <Sections>
   <Section Name="None">
    <Member Name="MyUDT1Member1" Datatype="bool"/>
    <Member Name="MyUDT1Member2" Datatype=&quot;myudt1&quot;>
     <Sections...
```

Remarque

Syntaxe pour les types de données personnalisés d'éléments

Si le type de données personnalisé d'un élément présente une mauvaise syntaxe dans le fichier d'importation pour types de données utilisateur, une exception est déclenchée.

Assurez-vous que &quot est utilisé pour noter les types de données personnalisés.

Code du programme

Pour importer un type de données utilisateur, modifiez le code de programme suivant :

```
//Imports user data type
private static void ImportUserDataType(PlcSoftware plcSoftware)
{
    string exportFileName = "ExportedPlcType.xml";
    string dirPathImport = @"C:\OpennessSamples\Import";
    string fullFilePath = Path.Combine(dirPathImport, exportFileName);
    PlcTypeComposition types = plcSoftware.TypeGroup.Types;
    IList<PlcType> importedTypes = types.Import(fullFilePath, ImportOptions.Override);
}
```

Voir aussi

Importation de données de configuration (Page 211)

Bibliothèques standard (Page 34)

8.4.5 Exportation de données au format OPC UA XML

Conditions requises

- L'application Openness est connectée à TIA Portal.
  Voir Etablissement d'une connexion au portail TIA (Page 69)
- Un projet est ouvert.
  Voir Ouvrir un projet (Page 90)
- L'API n'est pas en ligne.

Utilisation

Vous pouvez exporter des données API dans un fichier au format OPC UA XML en appelant une action sur Openness API. En paramètre d'entrée de l'action, vous avez besoin d'un chemin de répertoire absolu dans lequel sera enregistré le fichier XML.

Code de programme

Pour lancer le service `OPCUAExportProvider`, modifiez le code de programme suivant :

```
Project project = ...;
OPCUAExportProvider opcUAExportProvider = project.HwExtensions.Find("OPCUAExportProvider")
as OPCUAExportProvider;
```

Pour lancer l'exportation, modifiez le code de programme suivant :

```
Siemens.Engineering.HW.DeviceItem plc = ...;
opcUAExportProvider.Export(plc, string.Format(@"D:\OPC UA export files\{0}.xml",
plc.Name));
```

Les modifications les plus importantes dans Openness V14

9

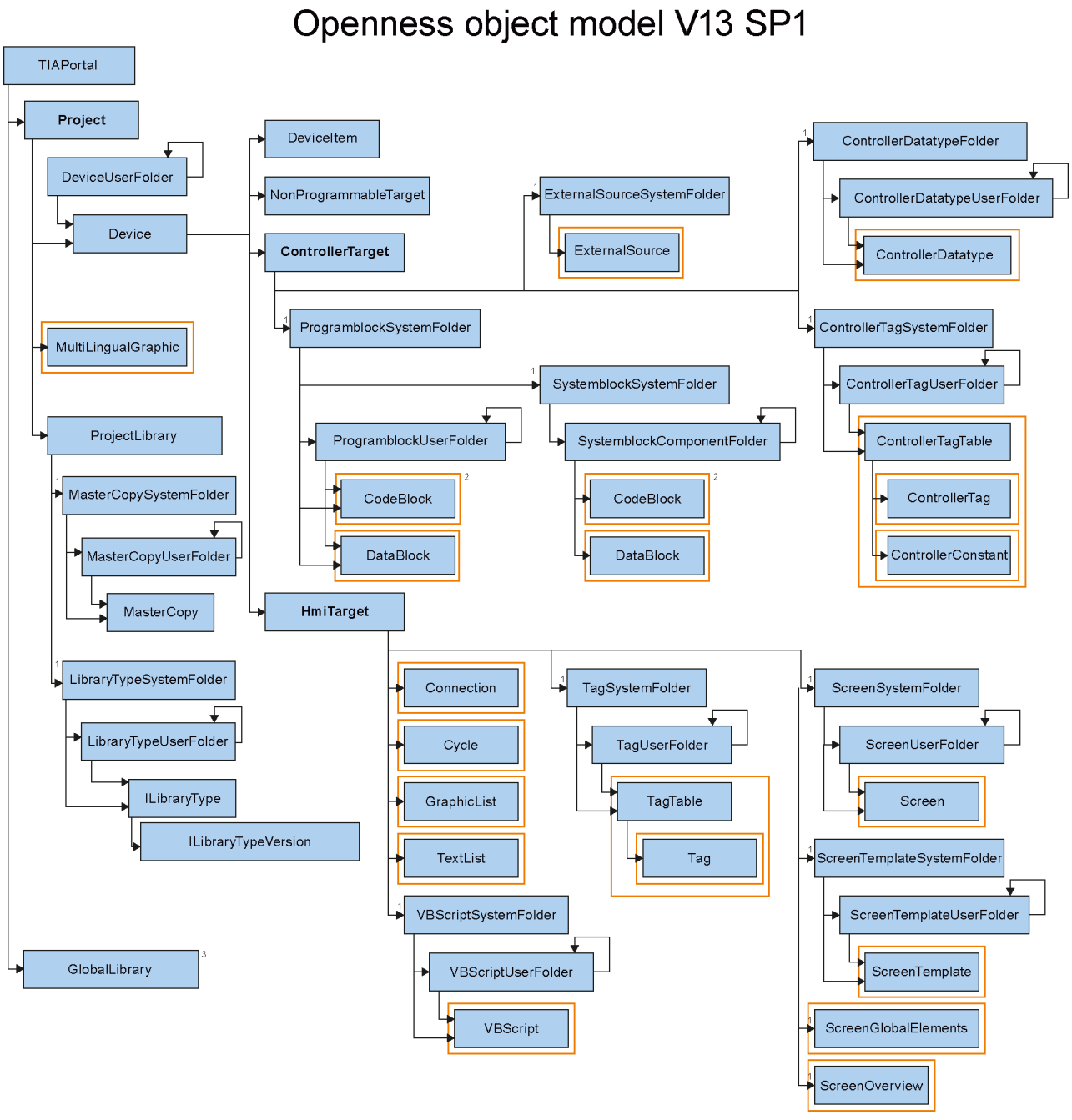
9.1 Principales modifications du modèle d'objet

Modèle d'objet de TIA Portal Openness V13 SP1 et plus ancien

Pour permettre une comparaison entre l'ancien et le nouveau modèle objet d'Openness, le diagramme ci-dessous décrit le modèle objet de TIA Portal V13 SP1.

Remarque

Le modèle objet décrit dans le diagramme est déconseillé. Vous trouverez des informations sur le modèle objet d'Openness V14 sous Modèle d'objet Openness V14 (Page 45)


Openness object model V13 SP1
TIAPortal
Project
DeviceUserFolder
Device
DeviceItem
NonProgrammableTarget
ControllerTarget
ExternalSourceSystemFolder
ExternalSource
ControllerDatatypeFolder
ControllerDatatypeUserFolder
ControllerDatatype
MultiLingualGraphic
ProgramblockSystemFolder
ControllerTagSystemFolder
SystemblockSystemFolder
ControllerTagUserFolder
ProjectLibrary
ProgramblockUserFolder
SystemblockComponentFolder
ControllerTagTable
MasterCopySystemFolder
CodeBlock
CodeBlock
ControllerTag
MasterCopyUserFolder
DataBlock
DataBlock
ControllerConstant
MasterCopy
HmiTarget
LibraryTypeSystemFolder
Connection
TagSystemFolder
ScreenSystemFolder
LibraryTypeUserFolder
Cycle
TagUserFolder
ScreenUserFolder
ILibraryType
GraphicList
TagTable
Screen
ILibraryTypeVersion
TextList
Tag
ScreenTemplateSystemFolder
VBScriptSystemFolder
ScreenTemplateUserFolder
GlobalLibrary
VBScriptUserFolder
ScreenTemplate
VBScript
ScreenGlobalElements
ScreenOverview

9.2 Avant la mise à niveau d'une application vers Openness V14

Application

Les paramètres suivants doivent être modifiés avant de mettre à niveau une application vers Openness V14 :

1. Les renvois vers l'API V14 doivent être adaptés en complétant les API Openness suivants :
   - `Siemens.Engineering`
   - `Siemens.Engfineering.Hmi`
2. Mettre à niveau le .Net-Framework de Visual Studio vers la version 4.6.1.
3. Actualisez la méthode AssemblyResolve en modifiant le nouveau chemin d'installation de TIA Portal.
   - Si vous travaillez à partir du fichier d'enregistrement, modifiez la nouvelle clé comme dans l'exemple suivant :
     `"HKEY_LOCAL_MACHINE\SOFTWARE\Siemens\Automation\_InstalledSW\TIAP14\TIA_Opns\..."`
   - Si vous travaillez avec le fichier de configuration d'application, adaptez les chemins au nouveau chemin d'installation.

9.3 Principales modifications de chaîne de caractères

Introduction

Les modifications suivantes ont été apportées à Openness V14 et il se peut que cela ait des effets sur vos applications existantes :

Modification	Adaptation nécessaire du code de programme
Les méthodes de compilation ont été modifiées.	Modifiez les méthodes de compilation comme dans l'exemple suivant : • Openness V13 SP1 (obsolète) : `controllerTarget.Compile(CompilerOptions.Software, BuildOptions.Rebuild);` • Openness V14: `plcSoftware.GetService<ICompilable>().Compile();`
De nouveaux espaces de noms ont été ajoutés.	1. Ajoutez les instructions d'espaces de noms suivantes : `Siemens.Engineering.SW.Blocks;` `Siemens.Engineering.SW.ExternalSources;` `Siemens.Engineering.SW.Tags;` `Siemens.Engineering.SW.Types;` 2. Supprimez l'instruction d'espace de nom `using ControllerTarget = Siemens.Engineering.HW.ControllerTarget.` 3. Compilez l'application.
`ControllerTarget` a été remplacé par `PlcSoftware` et la fonctionnalité a été modifiée dans certains cas.	1. Vérifiez les exemples de code dans la documentation faisant partie de votre fonctionnalité d'application. 2. Actualisez le code du programme de votre application Openness d'après l'exemple suivant : – Openness V13 SP1 (obsolète) : `ControllerTarget controllerTarget = deviceItem as ControllerTarget` – Openness V14: `PlcSoftware plcSoftware = deviceItem.GetService<SoftwareContainer>().Software as PlcSoftware` 3. Compilez l'application.

Modification	Adaptation nécessaire du code de programme

Des objets ont été remplacés.	1. Recherchez et remplacez les objets suivants :
	`DeviceUserFolderAggregation = DeviceUserGroupComposition`
	`DeviceFolders = DeviceGroups`
	`DeviceUserFolder = DeviceUserGroup`
	`ProgramblockSystemFolder = PlcBlockSystemGroup`
	`ProgramblockUserFolder = PlcBlockUserGroup`
	`IBlock = PlcBlock`
	`ControllerDatatypeSystemFolder = PlcTypeSystemGroup`
	`ControllerDatatypeUserFolder = PlcTypeUserGroup`
	`ControllerDatatype = PlcType`
	`ControllerTagSystemFolder = PlcTagTableSystemGroup`
	`ControllerTagUserFolder = PlcTagTableUserGroup`
	`ControllerTagTable = PlcTagTable`
	`ControllerTag = PlcTag`
	`ControllerConstant = PlcConstant`
	`ExternalSourceSystemFolder = PlcExternalSourceSystemGroup`
	`ExternalSource = PlcExternalSource`
	`IOnline = OnlineProvider`
	`ILibraryType = LibraryType`
	2. Compilez l'application.

Les agrégations ont été remplacées par des compositions.	1. Remplacez chaque `Aggregation` dans votre code par `Composition` comme dans les exemples suivants :
	`ProjectAggregation = ProjectComposition`
	`IDeviceAggregation = IDeviceComposition`
	`TagTableAggregation = TagTableComposition`
	`CycleAggregation = CycleComposition`
	`GraphicListAggregation = GraphicListComposition`
	`TextListAggregation = TextListComposition`
	`ConnectionAggregation = ConnectionComposition`
	`MultiLingualGraphicAggregation = MultiLingualGraphicComposition`
	`UpdateCheckResultMessageAggregation = UpdateCheckResultMessageComposition`
	2. Compilez l'application.

Les dossiers ont été remplacés par des groupes dans toutes les relations, sauf pour concerne les pupitres opérateurs.	1. Excepté les sections de code concernant les pupitres opérateurs, remplacez chaque `Folder` dans votre code de programme par `Group`. 2. Compilez l'application.

Modification	Adaptation nécessaire du code de programme
La méthode `GetAttributeNames` a été remplacée par la méthode `GetAttributeInfos` .	1. Pour déterminer les attributs, utilisez `IList<EngineeringAttributeInfo> IEngineeringObject.GetAttributeInfos(AttributeAccessMode attributeAccessMode); .` 2. Compilez l'application. Pour plus d'informations, référez-vous à Déterminer la structure et les attributs de l'objet (Page 98).
La méthode `Close` de fermeture d'un projet a été modifiée.	1. Remplacez `project.Close(CloseMode.PromptIfModified);` par `project.Close();`. 2. Compilez l'application. Pour plus d'informations, référez-vous à Fermer un projet (Page 158).

L'accès simultané a été remplacé par l'accès exclusif et des transactions exclusives.	1. Remplacez l'accès simultané par l'accès exclusif et des transactions exclusives comme dans les exemples suivants :
	– Openness V13 SP1 (obsolète) :
	`tiaProject.StartTransaction("Reseting project to default");`
	`...`
	`tiaProject.CommitTransaction();`
	– Openness V14:
	`//Use exclusive access to avoid user changes`
	`ExclusiveAccess exclusiveAccess = tiaPortal.ExclusiveAccess();`
	`...`
	`exclusiveAccess.Dispose();`
	`//Use transaction to be able to rollbank changes:`
	`Transaction transaction = exclusiveAccess.Transaction(tiaProject, "Compiling device");`
	`transaction.CommitOnDispose();`
	2. Compilez l'application.
	Pour plus d'informations, voir Exclusive access (Page 85) et Traitement des transactions (Page 87).

Modification	Adaptation nécessaire du code de programme

L'accès en ligne à la CPU a été modifié	1. Modifiez l'accès en ligne à la CPU comme dans les exemples suivants :
	– Openness V13 SP1 (obsolète) :
	`((IOnline)controllerTarget).GoOffline();`
	– Openness V14:
	`((DeviceItem) plcSoftware.Parent.Parent).GetService<Onlin eProvider>().GoOffline();`
	2. Compilez l'application.

La configuration matérielle a été modifiée	1. Modifiez la configuration matérielle : `Device.Elements = Device.Items` 2. Supprimez les propriétés matérielles suivantes : – `Device.InternalDeviceItem` – `Device.SubType` 3. Compilez l'application.

Voir aussi

Traitement des exceptions (Page 193)

Nouveautés d'Openness V14 (Page 13)

Etablissement d'une connexion au portail TIA (Page 69)

9.4 Importation de fichiers créés avec Openness V13 SP1 et des versions antérieures

Application

Si vous essayez d'importer des fichiers qui ont été générés avec Openness V13 SP1 ou une version antérieure, une exception d'incompatibilité est déclenchée. Motif : des modifications dans les variables IHM et les écrans IHM. Les tableaux suivants montrent les principales modifications d'attributs. Vous trouverez des informations détaillées dans le chapitre "Créer les vues > Utilisation des objets et de groupes d'objets > Utilisation des objets > Configuration de plages" dans l'aide en ligne de TIA Portal :

Modifications de variables IHM

Le tableau suivant montre les principales modifications d'attributs de variables IHM :

Attributs supprimés	Attributs ajoutés
RangeMaximumType	LimitUpper2Type.
RangeMaximum	LimitUpper2.
RangeMinimumType	LimitLower2Type.
RangeMinimum	LimitLower2.
	LimitUpper1Type
	LimitUpper1
	LimitLower1Type
	LimitLower1

Modifications d'éléments de vue IHM

Le tableau suivant montre les principales modifications d'attributs de curseur :

Attributs supprimés	Attributs ajoutés
	RangeLower1Color
	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled
	ScalePosition
	ShowLimitLines
	ShowLimitMarkers
	ShowLimitRanges

Le tableau suivant montre les principales modifications d'attributs de plage de mesure :

Attributs supprimés	Attributs ajoutés
DangerRangeColor	RangeLower1Color
DangerRangeStart	RangeLower1Enabled
DangerRangeVisible	RangeLower2Color
WarningRangeColor	RangeLower2Enabled
WarningRangeStart	RangeNormalColor
WarningRangeVisible	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper1Start
	RangeUpper2Color
	RangeUpper2Enabled
	RangeUpper2Start

Le tableau suivant montre les principales modifications d'attributs de bargraphe :

Attributs supprimés	Attributs ajoutés
AlarmLowerLimitColor	RangeLower1Color
AlarmUpperLimitColor	RangeLower1Enabled
	RangeLower2Color
	RangeLower2Enabled
	RangeNormalColor
	RangeNormalEnabled
	RangeUpper1Color
	RangeUpper1Enabled
	RangeUpper2Color
	RangeUpper2Enabled

Index

F

G

H

I

L

M

O

P

R

S

T

V