Gérer une application répartie avec Measurement Studio 6

Grégory VOIRIN

Ingénieur d'application

Plan de la présentation

- Applications réparties
- De COM à ActiveX
- L'assistant serveur ActiveX
- L'assistant contrôleur ActiveX
- La configuration DCOM
- Démonstration
- Questions

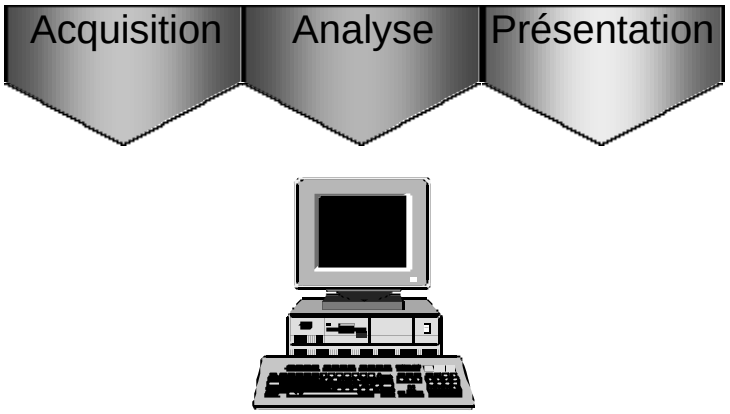
ni.com/france

NATIONAL INSTRUMENTS

Après avoir vu les différentes motivations qui peuvent orienter un développement vers une solution répartie, nous essayerons de faire un tour d'horizon des technologies ActiveX : leur but, leur structure et le vocabulaire associé.

La deuxième partie de cette présentation nous permettra de parcourir les étapes indispensables à l'élaboration d'une application répartie avec LabWindows/CVI. Après avoir parcouru les différents assistants, nous verrons un exemple d'application orientée mesure qui a été entièrement développée avec CVI 6.

Système de mesures

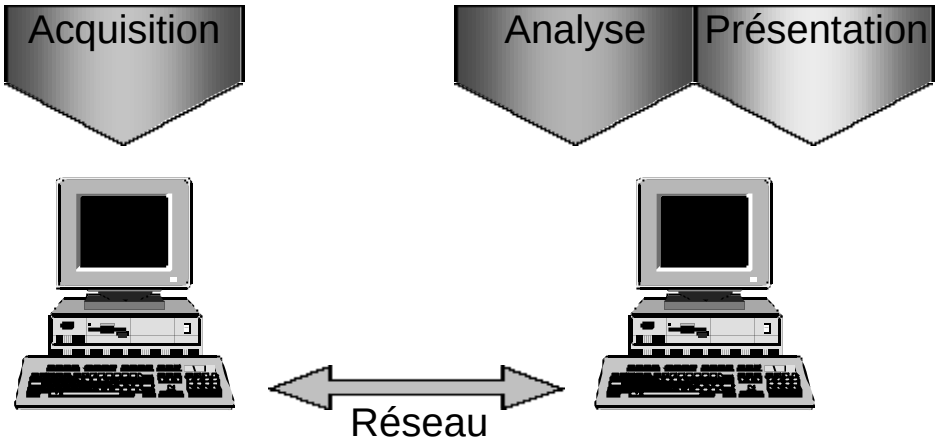


ni.com/france

Un système de mesures doit remplir trois fonctions principales : l'acquisition des données, l'analyse (traitement sur les données brutes afin d'en extraire les informations utiles) puis présentation des résultats. Traditionnellement, ces trois fonctionnalités étaient regroupées en un seul et même logiciel, tournant sur un poste informatique unique. L'évolution des PC était suffisamment rapide et importante pour que les performances de la plus grande partie des systèmes soient satisfaisantes. Toutefois, les cartes d'acquisitions permettent d'obtenir de plus en plus d'informations et les systèmes supportent un plus grand nombre de cartes. Si l'on ajoute à cela l'exigence croissante en matière de traitement et l'impossibilité de renouveler un parc informatique à chaque évolution, les capacités de calculs d'un ordinateur peuvent paraître trop limitées.

La solution la plus évidente est donc de faire coopérer les différentes machines disponibles, en s'appuyant sur la technologie des réseaux modernes.

Système de mesures réparti

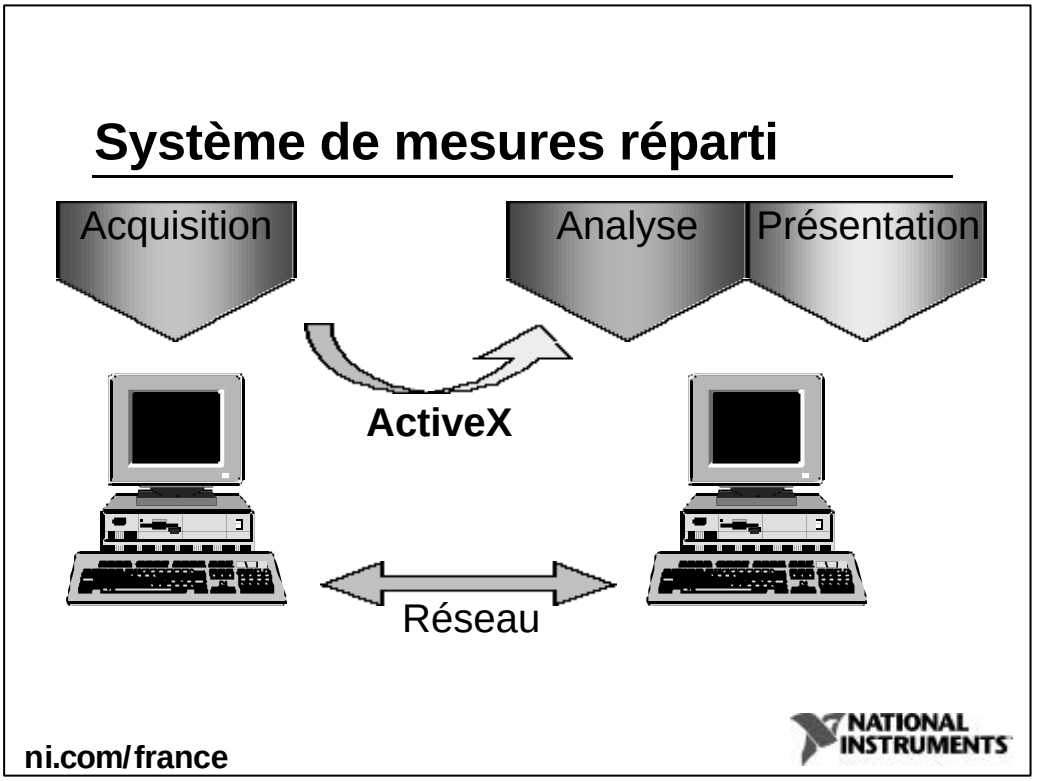


ni.com/france

NATIONAL INSTRUMENTS

Dans ce but, les systèmes de mesures sont aujourd'hui souvent répartis sur plusieurs postes informatiques. Ainsi les données peuvent être présentées là où elles sont attendues et non plus systématiquement là où elles sont acquises. De plus, les traitements nécessaires pour le passage de données brutes à une forme exploitable et intelligible sont de plus en plus complexes, longs et nécessitent de plus en plus de ressources matérielles. La répartition permet alors de ne pas altérer les performances du système d'acquisition qui n'a plus à prendre en charge l'ensemble des fonctionnalités du système d'acquisition.

Toutefois, la mise en œuvre de tels réseaux a longtemps été un obstacle à la réalisation d'applications réparties.



Même s'il a toujours été possible de développer et de déployer ce type d'applications, il fallait s'orienter vers des solutions propriétaires. La méthode n'était certes pas très compliquée :

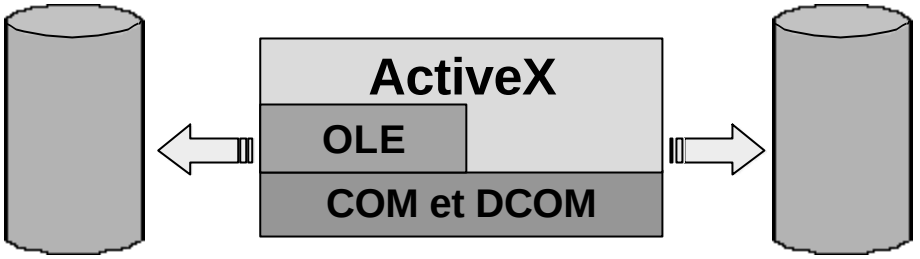
•définition d'un protocole propriétaire basé sur TCP/IP

•développement d'un serveur encapsulant (et complétant) un serveur TCP

•écriture d'un ou plusieurs client(s)

Toutefois, il apparaissait rapidement certaines limites. La moindre évolution du protocole nécessitait la mise à jour du code du serveur et de tous les clients, d'où une augmentation des coûts de maintenance de l'application. Il était également impossible d'interfacer en réseau ou même en local une application dont on ne possédait pas le code source, ce qui est le cas de l'ensemble des logiciels commerciaux. Il manquait donc une architecture simple et standardisée permettant de voir les différentes applications communiquer entre elles sans avoir à se soucier du protocole sous-jacent. C'est pour répondre à cette demande que Microsoft présenta la technologie ActiveX.

COM, DCOM, OLE et ActiveX



- Un protocole inter-applications indépendant du langage de développement
- Transparence du réseau

ni.com/france

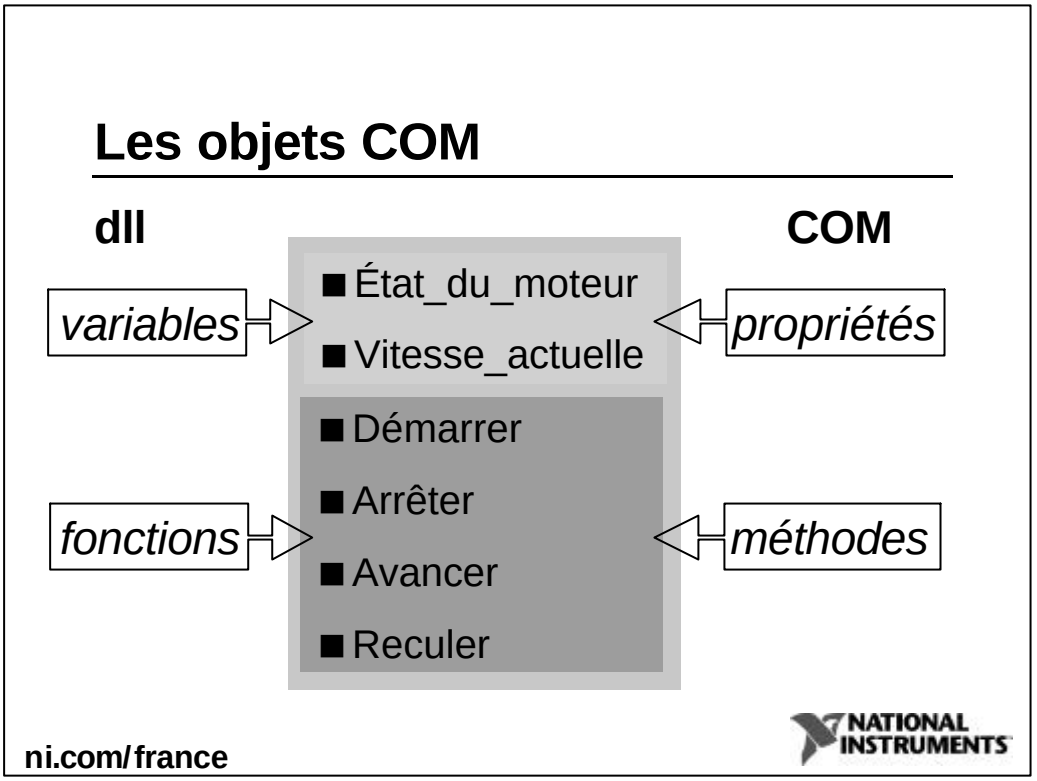
NATIONAL INSTRUMENTS

Les début de la communication inter-applications : COM

Une application COM (Component Object Model) est constituée d'un ou plusieurs objets COM, chacun d'entre eux exportant une ou plusieurs interfaces. Cette notion permet d'étendre la technologie objet à des applications développées à l'aide d'un langage n'intégrant pas cette philosophie (par exemple le C). Un objet COM est donc constitué de propriétés (des données) et de méthodes (des fonctions). Les interfaces sont en fait des ensembles et sous-ensembles de propriétés et méthodes que l'on veut rendre accessibles depuis une autre application.

L'évolution naturelle : ActiveX

Fort du succès de cette architecture, Microsoft a introduit DCOM (Distributed COM) qui permet d'étendre l'interconnectivité des composant COM au travers d'un réseau de façon totalement transparente et standard. Après avoir complété la couche de liaison inter-application avec OLE (Object Linking Embedded) Microsoft regroupa toute ces spécifications technologiques sous l'appellation ActiveX.



Derrière cette terminologie se cachent des concepts traditionnels de l'informatique. Réfléchissons par analogie : considérons une dll chargée de simuler une voiture. Elle pourrait exporter :

quatre fonctions :

- Démarrer
- Arrêter
- Avancer
- Reculer

et deux variables :

- État du moteur
- Vitesse actuelle

Il est tout à fait possible de créer un objet COM pour simuler cette même voiture. Les fonctions exportées s'appelle alors des **méthodes** et les variables des **propriétés**.

L'ensemble de ces méthodes et propriétés se nomme alors une interface.

La différence fondamentale avec une dll est alors que les variables ne sont pas accessibles directement : seules les méthodes permettent de consulter ou modifier leur valeur.

Les interfaces

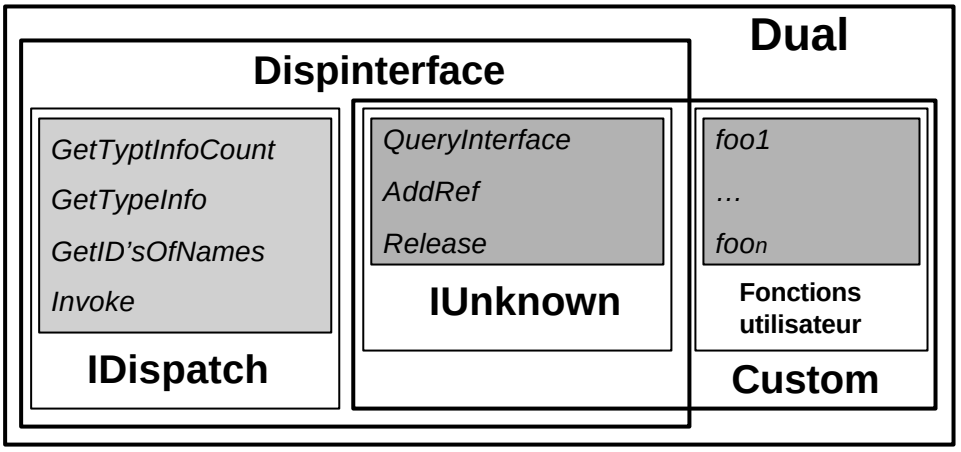
- Tous les utilisateurs n'accèdent pas aux mêmes fonctionnalités
- Organisation logique en fonction des besoins de chacun

Imaginons à présent que cette même dll doive également être destinée à un utilisateur qui ne s'intéresse qu'à la boîte de vitesse de la voiture. Il nous faudrait ajouter trois fonctions (Monter_un_rapport, Descendre_un_rapport, Mettre_au_point_mort) et une variable (Rapport_actuel). Notre dll devrait donc exporter en même temps les sept fonctions et les trois variables.

Un objet COM nous permet d'organiser plus logiquement les propriétés et les méthodes en les groupant selon l'utilisateur auquel elles s'adresse. Ces groupes s'appellent les **interfaces**, et chaque objet peut posséder une ou plusieurs interfaces.

Pour des raisons de standardisation, une interface doit, en plus de ces fonctions spécifiques, être munie de fonctions génériques liées à la gestion d'ActiveX. Le codage de celles-ci est assez routinier et nécessite un code robuste basé sur le système d'exploitation : c'est à ce stade que les différents assistants de CVI 6 interviennent.

Les interfaces disponibles



ni.com/france

NATIONAL INSTRUMENTS

Par souci de compatibilité, et du fait de l'évolution par couches du modèle ActiveX, Microsoft a appliqué le concept d'héritage (et même d'héritage multiple) aux interfaces disponibles.

En effet, si l'interface d'origine IUnknown s'est vu remplacée par l'interface IDispatch - qui apparaissait plus « conviviale »- ce remplacement s'est rapidement transformé en complémentarité pour donner naissance aux trois grands types d'interfaces utilisées aujourd'hui : Custom, Dispinterface et Dual.

L'amélioration des interfaces a considérablement simplifié l'accès aux « services » spécifiques de l'objet. Iunknown n'autorisait qu'une procédure d'appel des méthodes : un appel à la fonction QueryInterface retournait un tableau de pointeurs sur les méthodes que l'on appelait donc par adresse …

Cet exercice de style n'est aujourd'hui plus indispensable et l'assistant de CVI 6 masque les détails d'implémentation de ces techniques complexes.

L’assistant serveur ActiveX de CVI 6

- Génération automatique de code
- Génération du prototype des fonctions à écrire
- Génération d’un exemple de fonction WinMain adaptée à la configuration choisie
- Configuration intuitive

ni.com/france

NATIONAL INSTRUMENTS

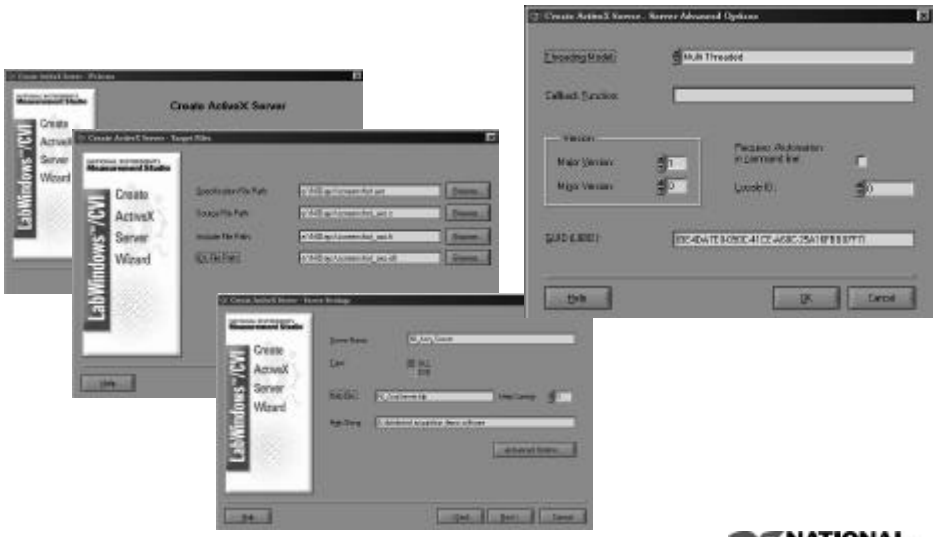
Les problèmes liés à la standardisation et à la gestion du réseau sont donc à la charge d’ActiveX, mais il reste un inconvénient majeur : le surcroît de code à écrire …

Une grande partie du développement d’un objet COM est malheureusement le codage de tâches génériques et la programmation système. Ce travail est non seulement routinier mais en plus la mise en œuvre de ces fonctionnalités est cruciale pour le bon fonctionnement du programme.

Pour pallier cet inconvénient, CVI 6 propose un assistant qui génère la plus grande partie du code (fonctions génériques et d’enregistrement au sein du système d’exploitation par exemple) et propose un modèle pour la partie spécifique à votre application. Il est donc à présent possible de développer un système réparti aussi simplement qu’un système de mesures traditionnel.

Notons qu’il est de plus très facile grâce à cet assistant de transformer une application ou une dll existante en serveur ActiveX. En effet, le code généré automatiquement est totalement indépendant du code contenu dans le projet existant.

Le choix de la configuration générale d'un serveur ActiveX



ni.com/france

NATIONAL INSTRUMENTS

La première phase de création d'un serveur ActiveX s'intéresse exclusivement à la configuration générale du serveur. Il ne faut toutefois négliger aucune option car chacune peut avoir une très nette influence sur le fonctionnement de l'application.

Cette étape permet de préciser, entre autre, l'ensemble des fichiers (chemin et noms) qui devront être générés. Cela inclut le fichier .axs (informations de configuration du serveur), un fichier .idl (Interface Definition Langage) destiné au compilateur Microsoft MIDLC au moment de la génération finale du serveur ActiveX, ainsi qu'un fichier .h de prototypage des fonctions et un fichier .c qui contient le code générique. Celui-ci est le seul que nous serons amenés à modifier.

Notons que les options avancées donnent accès à tous les modèles mono-thread ou multi-threads définis par Microsoft pour le standard ActiveX. Il existe également la possibilité de définir une callback qui se déclenchera au démarrage et à la fermeture du serveur.

Les options sont très nombreuses et je vous conseille donc de télécharger le document (au format pdf) qui est disponible sur le site National Instruments. Celui-ci est disponible en français.

L’édition du serveur

- Un serveur doit contenir au moins un objet
- Un objet doit exporter au moins une interface
- L’assistant permet de dissocier la sauvegarde de la génération de code

ni.com/france

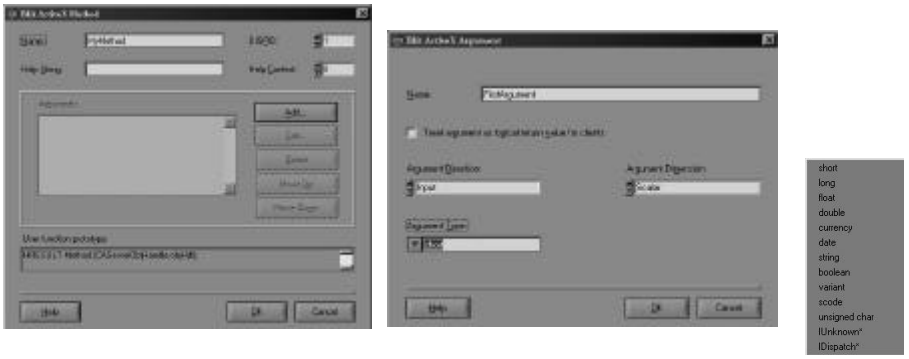
NATIONAL INSTRUMENTS

Après la configuration générale du serveur, il faut s’intéresser à son contenu : les objets et leurs interfaces.

Lorsque l’on souhaite créer un objet, en programmation comme dans le monde réel, il est toujours plus naturel de s’intéresser d’abord aux moyens que l’utilisateur aura à sa disposition pour interagir avec cet objet. Dans notre cas, c’est la phase de conception de l’interface. Il nous faudra ensuite identifier clairement les « services » que l’objet pourra fournir à ses utilisateurs (il s’agit de ses méthodes) ainsi que les « éléments » qu’il devra contenir (ce sont les propriétés). Ce n’est qu’à ce stade qu’il sera enfin possible de « construire » l’objet, c’est-à-dire de générer le code.

Cette dernière étape est considérablement simplifiée par CVI 6. En effet, LabWindows génère tous les prototypes des fonctions et le corps de celles qui sont génériques. Vous n’avez plus qu’à développer les fonctions spécifiques à votre serveur. De plus, la phase de conception étant généralement la plus longue, et comme il est inutile de générer un code incomplet, l’assistant permet de dissocier la sauvegarde des spécifications et la génération de code.

Les méthodes



Prototypage rapide et intuitif des méthodes

ni.com/france

NATIONAL INSTRUMENTS

Après avoir choisi d'insérer une méthode, il suffit de préciser son nom et de configurer ses arguments ainsi :

•le nom : il doit respecter les règles communes à tous les langages pour rester indépendant

•le type : la sélection au sein de la liste évite toute erreur

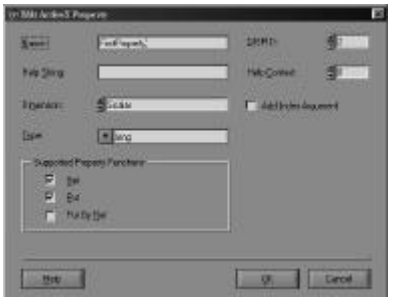
•la dimension (scalaire, tableau à une ou deux dimensions)

•la direction (entrée ou sortie de la fonction) : c'est ce paramètre qui déterminera si le passage de l'argument est fait par valeur ou par référence.

La sélection de l'option « Treat argument as logical return value » transforme le paramètre en valeur de retour de la fonction : il ne peut donc y avoir qu'un seul argument avec cette option dont la direction sera output et qui devra toujours être le dernier argument de la liste.

Les propriétés

- Typage simplifié
- Choix de la dimension
- Choix d'un support pour les fonctions d'accès



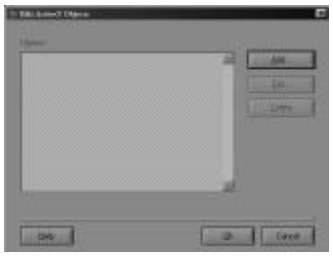
ni.com/france

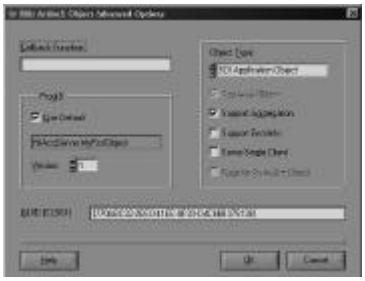
Basé sur le même principe de sélection par menu déroulant et cases à cocher, le typage des propriétés est tout aussi simple : cela permet de garantir un choix de type compatible entre langages de développement et de maintenir une notion forte de standard.

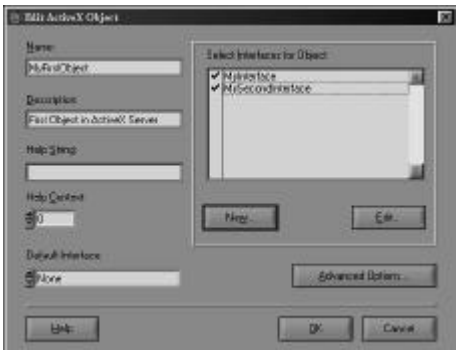
La sélection du support des différentes fonctions d'accès influence la configuration des clients qui pourront accéder aux données. Pour une compatibilité étendue, il peut être judicieux de cocher les trois cases. Pour de plus amples informations, vous pouvez vous reporter au chapitre de configuration d'un client ActiveX.

L'option Add Index Argument permet d'ajouter un paramètre de type entier sur 32 bits aux fonctions d'accès aux propriétés. Ce paramètre pourra être utilisé comme un index pour l'accès aux données à une ou deux dimensions (tableaux).

Les objets







ni.com/france

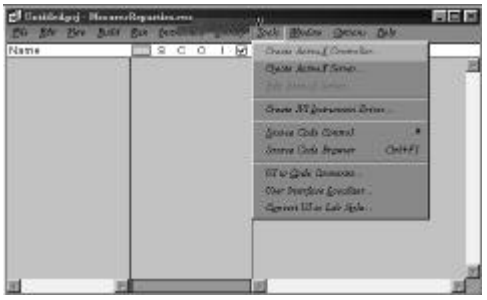
CVI 6 permet la définition d'une fonction callback associée au serveur ActiveX. Si celle-ci est implémentée, elle est appelé lors des deux évènements suivants : CA_SERVER_EVENT_OBJECT_CREATE (création de l'objet) et CA_SERVER_EVENT_OBJECT_DESTROY (destruction de l'objet).

Le type de l'objet doit également être précisé sans perdre de vue que le seul type supporté par les serveurs ActiveX sous forme de dll est le type Custom.

Il est enfin nécessaire de choisir si l'objet doit supporter l'agrégation, s'il peut ou non servir plusieurs clients en même temps et s'il doit être enregistré comme actif dans la Running Object Table.

L’Assistant pour la création de contrôleurs ActiveX

Génération d’un panneau de fonction et du code associé



ni.com/france

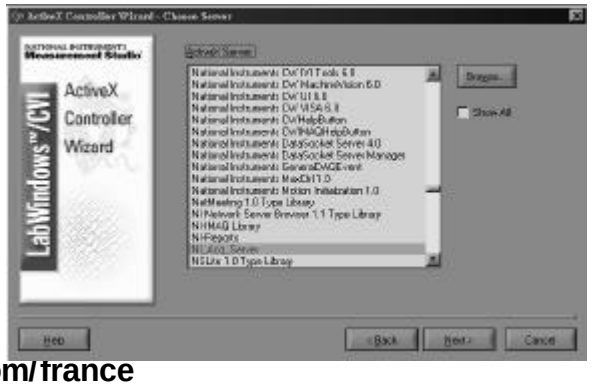
Le serveur ayant été créé, il nous faut à présent développer la partie cliente de notre application globale.

Le principal centre d’intérêt d’un développeur de système de mesures basé PC ne doit jamais être le codage d’appels systèmes ou de fonctions génériques répétitives. Dans ce souci de simplification d’accès aux technologies informatiques de pointe, LabWindows/CVI vous propose un assistant pour la création d’un driver d’instrument permettant de dialoguer avec un serveur ActiveX. En répondant à quelques questions élémentaires, le panneau de fonction et le code associé seront automatiquement générés et éventuellement intégrés au projet en cours.

Nous allons donc suivre pas à pas le processus permettant d’obtenir le driver d’instrument pour le serveur ActiveX.

Choix d’un serveur

- Le système d’exploitation connaît la liste de tous les serveurs ActiveX présents sur le poste
- Le serveur peut être enregistré directement par l’assistant



ni.com/france

NATIONAL INSTRUMENTS

La première étape consiste à choisir le serveur pour lequel on veut développer un client.

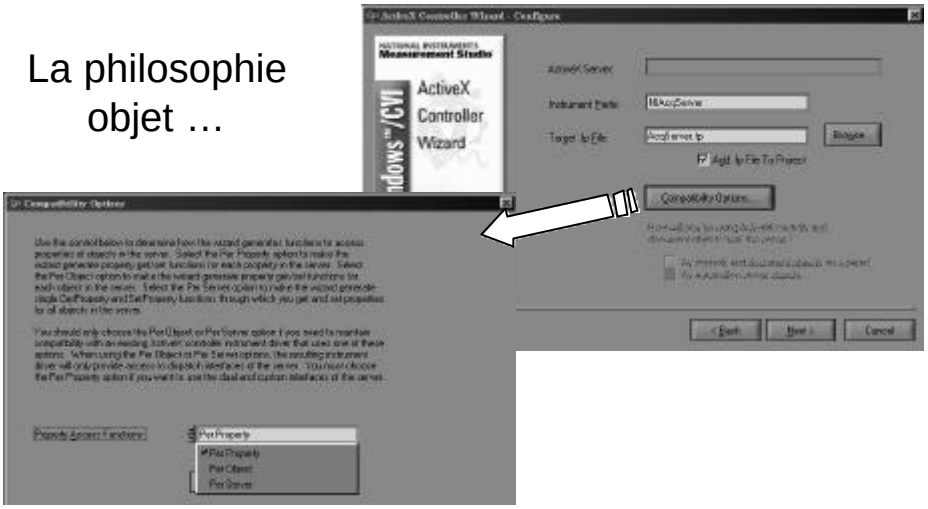
Au travers de la base des registres, le système d’exploitation connaît l’ensemble des serveurs ActiveX utilisables depuis un poste donné. Si le serveur a été développé avec CVI, il aura été enregistré automatiquement lors de la création de l’exécutable ou de la dll. La fonction d’enregistrement, qui modifie la base de registres est automatiquement générée et exécutée par CVI. Le nom du serveur apparaît donc dans la liste.

La checkbox Show All permet d’afficher les serveurs qui auraient été marqués comme cachés ou à accès restreint et qui n’apparaissent donc pas par défaut dans la liste des serveurs disponibles.

De plus, le bouton Browse permet de sélectionner le fichier librairie d’un objet serveur si celui-ci n’a pas été enregistré au préalable. Cette sélection aura pour effet d’enregistrer automatiquement ce serveur sur le poste et de faire apparaître son nom dans la liste des serveurs disponibles.

Le driver d'instrument

La philosophie objet …



ni.com/france

NATIONAL INSTRUMENTS

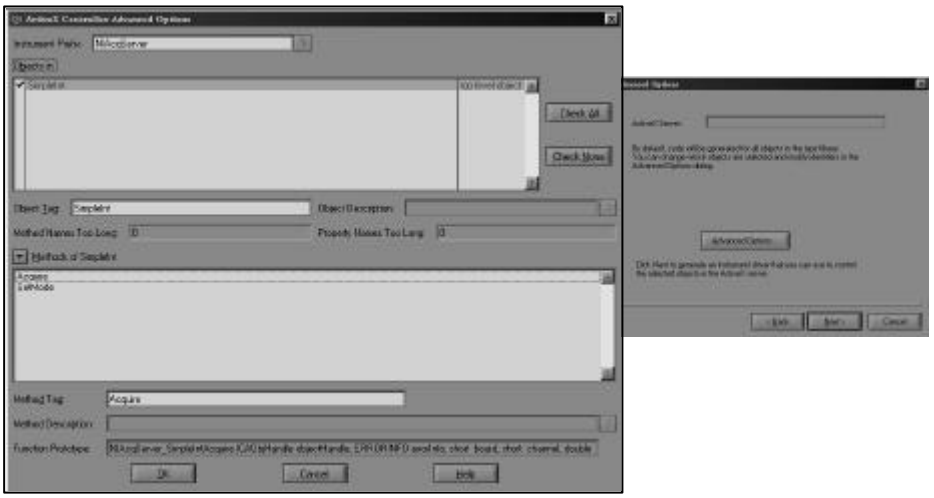
L'étape suivante permet de configurer toutes les options du driver d'instrument lui-même. Il faut choisir le préfixe qui sera utilisé pour le nom des fonctions ainsi que le nom et le chemin du fichier .fp correspondant. A ce stade, on peut également choisir d'ajouter le panneau de fonction au projet ouvert dans CVI.

Il faut ensuite préciser la stratégie à adopter pour l'accès aux propriétés du serveur. Trois possibilités s'offrent à vous :

- Par Propriété : l'assistant génèrera une fonction Get et une fonction Set par propriété. Exemple : SetChanel, GetScanRate …
- Par Objet : l'assistant génère une fonction Get et une function Set par objet, ces fonctions donnant accès à toutes les propriétés de l'objet. Exemple : GetAcquitionSystem (Chanel, …), GetAcquisitionSystem(ScanRate, …)
- Par serveur : l'assistant génère en tout deux fonctions (SetProperty et GetProperty) pour accèder à toutes les propriétés de tous les objets du serveur. Exemple : SetProperty(AcquisitionSystem, Chanel, …), GetProperty(AcquisitionSystem, ScanRate, …)

ATTENTION : Si vous utilisez une interface Dual ou Custom, vous ne pouvez choisir que la compatibilité par propriété car les deux autres méthodes d'accès ne sont compatibles qu'avec l'interface IDispatch.

Sélection des objets



ni.com/france

La fenêtre **Advanced Options** permet de sélectionner clairement les objets qui seront utilisés par le driver d'instrument. Pour faciliter ce choix, lorsqu'un objet est sélectionné, il est possible de parcourir ses propriétés et ses méthodes. Vous constaterez que les informations de typage des méthodes sont présentes dans la rubrique **Function Prototype**, rubrique qui n'est bien sûr pas disponible lorsque l'on parcourt les propriétés.

Pour valider la présence d'un objet dans le driver d'instrument, il faut s'assurer que la ligne est cochée. Il est possible de cocher (ou de décocher) en une seule opération tous les objets d'une interface.

Le driver d'instrument est ensuite généré et il peut être utilisé pour développer la partie cliente de notre application. La phase de développement est à présent terminée.

L’enregistrement à distance.

- Pour un exécutable :

  Monserveur.exe /regserver

- Pour une dll :

  RegSrv32.exe Monserveur.dll

ni.com/france

NATIONAL INSTRUMENTS

Il reste malgré tout deux étape pour utiliser le client et le serveur sur deux postes différents.

Il faut tout d’abord enregistrer le serveur sur le poste client au même titre qu’il devait l’être sur le poste de développement de celui-ci. La méthode dépend ici de la nature du serveur.

Le serveur est un exécutable :

Dans une fenêtre MS-DOS (ou dans la fenêtre de dialogue Exécuter du menu Démarrer), il faut lancer l’exécutable depuis son emplacement réseau en ajoutant la paramètre /regserver

Le serveur est une dll :

Il faut lancer la commande RegSrv32.exe suivi du nom de la dll précédée de son chemin réseau complet.

Notre serveur est désormais visible et les informations nécessaires à une connexion distante sont disponibles sur le poste client.

Configuration DCOM

- Windows 98 ou Windows NT ?
- Sécurité et ActiveX
- Configuration Client et Serveur

ni.com/france

NATIONAL INSTRUMENTS

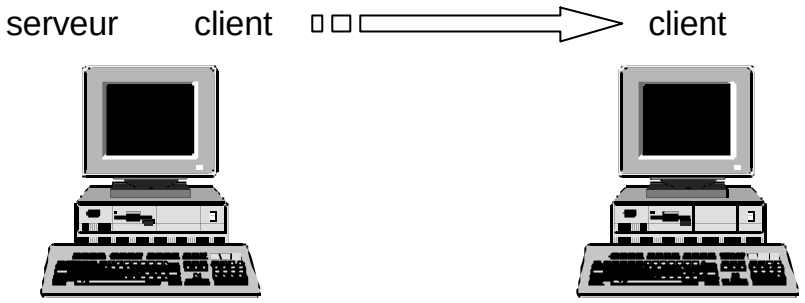
Si la gestion de réseau ne nécessite aucun codage spécifique, il faut toutefois configurer les différents droits d'accès. Pour cela on fait appel à l'utilitaire dcomcnfg.exe, qui liste tous les serveurs accessibles et permet la configuration de façon globale (tous les serveurs ont la même stratégie de sécurité) ou serveur par serveur.

Le choix du système d'exploitation est alors capital : la configuration DCOM d'un serveur sur une plate-forme autre que Windows NT ou 2000 relève souvent du tour de force…

Si l'on ajoute à cela l'impossibilité pour Windows 98 ou ME de lancer automatiquement le serveur quand un client tente de s'y connecter, le choix du système d'exploitation ne laisse aucun doute.

Le détail de configuration DCOM étant totalement lié au système d'exploitation, il est préférable de se reporter à la documentation Microsoft. Notons toutefois qu'il est impératif que le poste client et le poste serveur possèdent des configurations cohérentes et, pour éviter toute complication, identiques.

Démonstration



NATIONAL INSTRUMENTS

Questions et commentaires

ni.com/france

NATIONAL INSTRUMENTS