

Table des matières

Utiliser NetRemote Designer.....	2
Le fichier CCF.....	2
Les Templates.....	2
Les Bitmaps.....	2
Les Boutons.....	2
Les Frames.....	2
Envoi d'un CCF sur le Pocket PC.....	3
Touches matérielles (Hardkeys).....	3
Plugins.....	3
NetRemote Variables.....	3
Quelques variables prédéfinies.....	3
Afficher une variable.....	4
Manipuler une variable.....	4
Utiliser les variables pour contrôler l'état d'un élément.....	4
Afficher une image dans un élément.....	4
Les "Button Components".....	4
Les Frames animées.....	5
Scripts Lua.....	5
Exécuter un script à partir d'un Bouton.....	5
Déterminer l'état d'un élément.....	5
Variables Lua.....	6
Actions et Variables incluses.....	6
Actions.....	6
Variables.....	6
Plugins et Instances de Plugins.....	6
Les Plugins.....	7
NRBasic.....	7
AvidUtils.....	7
FlashPlayer.....	8
Generic.....	8
Girder.....	8
Infrared.....	8
MediaBridge.....	8
Wake On Lan.....	9
WebBrowser.....	9
Zoom.....	9
Les conteneurs de scripts Lua.....	9
Lua Library Reference.....	9
Notation.....	9
NetRemote Library.....	10
Fonctions de création, d'accès et de contrôle de variables NetRemote.....	10

Utiliser NetRemote Designer

Le fichier CCF

Un fichier ccf est composé de **Pages**, organisées en **Page Groups**, chacun correspondant en général à un appareil

Une page s'affiche à la fois en plein écran

Une page comporte des **Frames** et des **Boutons**

Une Frame peut contenir d'autres Frames et Boutons

Un bouton peut être pressé et déclencher une liste **d'Actions**

L'aspect visuel d'une Frame ou un bouton peut être une Bitmap, les frames non rectangulaires utilisent un fond transparent

Boutons spécialisés : barre de progression, boîtes de saisie, navigateur web, ...

Toutes les pages d'un ccf ont la même résolution (click-droit sur "System", "Properties")

Ne pas oublier de choisir le "File Format" : j'ai choisi 1.1, revision 1.4

Les Pages dans un Page Group partagent le même **Template**

Les Templates

Un Template permet de définir des parties communes qui n'ont pas besoin d'être répétées sur chaque page partageant ce Template : contrôle de volume, mute, logo, ...

Trois Templates : masterTemplate, deviceTemplate et macroTemplate

Chaque Template a un Overlay associé

Les éléments définis sur une page s'affichent par dessus les éléments du Template associé

Les éléments définis sur l'Overlay s'affichent par dessus les éléments de la page :

xTemplate Overlay > Page > xTemplate

Les Bitmaps

Les images utilisées dans les Frames et les Boutons sont référencées par un numéro d'ordre : pour changer le look d'un CCF, il suffit d'avoir un groupe de fichiers images avec le même ordonnancement (name-0.png, name-1.png, ...) et la même taille

Les Boutons

"Display>Element Name" peut être utilisé pour faire référence au bouton. Click-droit dans "Element Name" pour choisir un caractère dans la police Pronto

Deux apparences : **Button Up/Released** et **Button Down/Pressed**

On peut créer un second état par click-droit dans "Display>States", fournissant une seconde paire d'apparences

Les Frames

Une Frame peut contenir d'autres Frames et Boutons : elle se comporte alors comme une unité : pour ajouter une Frame ou un bouton à une Frame, on la sélectionne puis click-droit dans la Frame

Pour créer un tableau d'éléments identiques dans une Frame, on crée une instance de l'élément à répéter et on le positionne à l'endroit où doit se situer le premier élément du tableau. Click-droit sur la Frame,

"Properties>Position/Advanced", on indique le nombre d'éléments dans "#X" et "#Y", le premier index est spécifié dans "Start" (vers la droite et vers le bas)

La donnée "<LoopIndex>" dans "Element Name" est remplacée par la valeur de l'index quand le client est exécuté. La donnée "<^LoopIndex>" est remplacée par l'index du parent (la Frame qui contient l'élément)

Envoi d'un CCF sur le Pocket PC

Connecter le PPC au PC avec une connexion Active Sync. Sur le PC aller dans le répertoire **NRCCF** du mobile, et copier le fichier CCF dans ce répertoire, copier éventuellement le fichier lua attaché. Sur le client PPC chargé le CCF

Touches matérielles (Hardkeys)

Chaque Page a sa propre liste de Hardkeys, on peut ainsi avoir des actions différentes suivant la Page

- FarLeft :
- Left :
- Right :
- FarRight :
- ArrowUp :
- ArrowDown :
- ArrowLeft :
- ArrowRight :
- ArrowSelect :

Plugins

Les Plugins sont configurés dans le client, ce qui détermine les variables NR disponibles, les Actions utilisables, ...

- **NRBasic** : Actions et Variables liées à l'utilisation de NetRemote (manipulation de variables, contrôles écran et appareils, exécution d'application et de scripts Lua, ...
- **AvidUtils** : actions additionnelles incluant des "Buttons Components" tels qu'un editeur de texte, listes déroulantes, ...
- **WebBrowser** :
- **FlashPlayer** : Button Component qui joue des fichiers d'animation Macromedia Falsh
- **Infrared** : contrôle l'envoi de codes IR
- **Girder** : pour s'interfacer avec Girde à travers le réseau
- **MediaBridge** :
- **Zoom** :
- **Wake On Lan** :
- **Generic** :

NetRemote Variables

Une NR Variable a un nom et une valeur (chaîne de caractères ou image)

Un point dans le nom de la variable crée une structure dans Variable Inspector

Si une variable devant être interprétée comme un nombre n'est pas reconnue, elle prend la valeur 0

Pour une variable devant être interprétée comme un booléen, "1" représente **true** ou **yes**, "0" **false** ou **no**, une chaîne vide est interprétée comme false, sinon comme true

Une variable peut être mis-à-jour suite à un événement par NetRemote ou un de ses plugins

Quelques variables prédéfinies

- **<LoopIndex>** : index courant de l'élément
- **<^LoopIndex>** : index courant du parent de l'élément
- **<^>** : nom du parent de l'élément

On peut utiliser plusieurs "^" successifs pour remonter dans la parenté

Afficher une variable

On inscrit le nom de la variable dans le champ "Display>Element name" d'une frame ou d'un bouton entre {} : dans ce cas la variable est mis-à-jour chaque fois qu'elle change

Si la variable est mise entre <> elle est évaluée une seule fois, lors du chargement de la page

Manipuler une variable

Le plugin **NRBasic** fournit des actions pour manipuler les variables

- **Set variable to value** : crée la variable si elle n'existe pas
- **Add value to variable** : ajoute à la valeur courante
- **Toggle value of variable** : si "0" devient "1", sinon devient "0"
- **Clear image variable** : efface l'image
- **Wait for variable to equal a value** : diffère l'exécution de la liste d'actions jusqu'à ce que la variable atteigne une valeur donnée, ou un temps maxi est atteint
- **Stop all actions if the variable does not equal a value** : si la variable n'est pas égale à une valeur donnée, saute la suite des actions de la liste
- **Skip the next action if the variable does not equal a value** : si la variable n'est pas égale à une valeur donnée, saute l'action suivante dans la liste

Le plugin **AvidUtils** fournit les actions :

- **Set a variable after a delay** : positionne une variable après un délai donné, le répète un certain nombre de fois ou indéfiniment
- **Cancel a scheduled set of a variable** : annule la planification d'une variable

Utiliser les variables pour contrôler l'état d'un élément

Dans "Properties>Display" : on crée un état "State 1" dans "States" (click-droit dans la zone), on peut alors sélectionner une variable dans "Rules", si celle-ci est égale à la valeur indiquée, l'élément sera montré dans cet état, sinon dans l'état "default". On peut ainsi créer plusieurs états

Si "Hidden State" est coché, l'élément est caché dans l'état correspondant

Afficher une image dans un élément

Dans "Properties>Position/Advanced" : on peut indiquer la référence à une image dans "Image Variable", utiliser pour afficher une pochette de CD par exemple. Cette image est "poussée" par un server (par exemple Girder)

L'action **Clear Image Variable** efface l'image

Les "Button Components"

Ce sont des boutons avec un aspect visuel spécifique avec des spécificités interactives

On crée un Bouton normal, puis double-click "Position/Advanced", sélection dans la liste "Component"

Les actions d'un tel bouton sont prédéfinies

Le plugin **NRBasic** fournit :

- **Text Edit Box Component** : la valeur de la variable peut être modifiée interactivement
- **Slider Thumb Component** : implémente un curseur de contrôle (exemple : contrôle du volume audio), le bouton est contenu dans une frame qui définit les limites de son déplacement. Déplacer le curseur modifie la variable et inversement
- **Image Progress Bar Component** : montre une barre de progression, affichage graphique de la valeur d'une variable entre des limites
- **ActiveX Component** :

Fournit par le plugin **WebBrowser** :

- **Embedded Web Browser Component** :

Fournit par le plugin **FlashPlayer** :

- **Flash Player Component** :

Fournit par le plugin **AvidUtils** :

- **Dropdown List Components** : affiche une liste déroulante pour en sélectionner un élément
- **Text String Component** : permet l'affichage et l'édition des variables comme texte
- **Scrolling Text Component** : affiche le texte entré dans "Element Name" avec un scrolling automatique

Fournit par le plugin **Girder** :

- **Girder Mouse Mode Component** : affiche un pas souris virtuel pour contrôler la souris d'un PC sous Windows exécutant Girder
- Fournit par le plugin **MediaBridge** :
- **MediaBridge Component** :

Les Frames animées

C'est une frame qui est déplacée par rapport à sa page ou sa frame parent, de sa position initiale à une position finale. Pour créer une telle Frame :

1. Créer une Frame sur la Page ou la Frame parent, avec un nom unique, ajouter à cette Frame les éléments que l'on désire
2. Ajouter un Bouton de contrôle sur la Frame animée, sur le côté de la Frame opposé à la direction de l'animation
3. Insérer l'action "NRBasic:Lua Event" avec comme nom "SlideAuto"
4. Positionner la Frame à sa position quand elle est fermée ou cachée (seul le bouton de contrôle doit être visible)

En créant un bouton invisible sur la Frame animée, avec un nom spécial ("Element Name") on accède à des actions avancées :

- **TIMEOUT:nn** : la Frame se ferme automatiquement après nn secondes
- **CLOSE:x** : si x=*, ferme toutes les autres Frames animées ouvertes avec le même parent, si x est une liste de Frame animées (séparées par un espace), ces Frames sont fermées si elles sont ouvertes. Peut être utilisé pour implémenter des menus (quand un menu s'ouvre, celui ouvert doit se fermer)
- **OPEN:x** : pour lier des Frames ensemble (par exemple ouverture d'un rideau : une frame vers la gauche, une vers la droite)
- **LINK:x** : les Frames liées se déplacent dans le même sens que la Frame de contrôle

Si le Bouton de contrôle n'est pas sur la Frame animée, le code lua devra être **AutoSlide("framename")**

Voir **Element Animation Library** pour d'autres fonctionnalités

Scripts Lua

Quand un client charge un fichier CCF, il regarde dans le même répertoire la présence d'un fichier avec le même nom, mais avec l'extension lua, dans ce cas il le compile et l'exécute. Pour créer et modifier ce fichier Lua, click-droit sur System et "Edit Lua Script File"

Ce script peut initialiser des variables NR et installer des fonctions Lua

Exécuter un script à partir d'un Bouton

Le plugin NRBasic fournit :

- **Execute lua code (OnDown)** : code exécuté quand le bouton est pressé
- **Execute lua code (OnRelease)** : code exécuté quand le bouton est relâché
- **Lua Event** : code exécuté dans les deux cas, et répété s'il est maintenu pressé

Déterminer l'état d'un élément

Dans "Properties>Dispaly" on a créé un état "State 1", on sélectionne "Lua" dans "Rule", click sur "Edit Lua

Code", pour sélectionner cet état le code Lua doit retourner **true**, sion **false**

Si "Hidden State" est coché l'état fait l'élément invisible

Variables Lua

Différentes des Variables NetRemote, types **numbers**, **strings**, **boolean values**, **functions**, **tables**, **objects**.
Lua fournit des variables **Global** et **Local**

Actions et Variables incluses

Actions

- **Alias** : exécute la liste d'action du Bouton sélectionné (ce bouton est souvent situé sur une page qui ne sera jamais affichée, elle peut être cachée, une page macro)
- **Beep** : émet un son
- **Delay** : suspend l'exécution des actions de la liste durant un temps donné
- **Infrared** : à priori ne fait rien sauf si le plugin **Infrared** est disponible
- **Jump** : affiche la page indiquée (dernière action dans la liste), soit une page spécifiée, soit la précédente affichée (Back), ou la suivante/précédente dans la courante **Page Group** (**Scroll Down**, **Scroll Up**)

Variables

- **NRFULLVERSION** : affiche nom du produit et version
- **NRPLATFORM** / **NRCFG.Platform** : **WIN32** ou **WINCE**
- **NRVERSION** / **NRCFG.Version** : version
- **NR.ClockDate** : date courante dans le format spécifié dans "File>Properties>General" (chez le client)
- **NR.ClockTime** : heure courante dans le format spécifié dans "File>Properties>General" (chez le client)
- **NR.DeviceName** : nom du Page Group (Device) courant
- **NR.PanelName** : nom de la page courante (Panel)
- **NR.FarLeftName** / **NR.LeftName** / **NR.RightName** / **NR.FarRightName** : labels des touches matériel
- **NR.InstallDir** : chemin de l'exécutable client
- **NR.FILE.Name** : nom complet du fichier CCF en cours
- **NR.FILE.Path** : chemin du fichier CCF
- **NRCFG.DeviceHeight** / **NRCFG.DeviceWidth** : taille de l'écran du client
- **NRCFG.IsRegistered** : "1" si le client est enregistré, sinon "0"
- **NRCFG.ShapedWindowsDisabled** : "1" si "Shaped Windows" est désactivé dans "Files>Properties" du client (sur PC)

Plugins et Instances de Plugins

Une instance correspond à une exécution du plugin, plusieurs instances du même plugin peuvent s'exécuter. Pour gérer ces instances on dispose des actions suivantes :

- **Next plugin instance** :
- **Previous plugin instance** :
- **Set current plugin instance** :

Chaque **Plugin Type** a un **Type ID**, identique pour toutes les instances d'une même plugin, chaque instance a un **Instance ID** unique.

- **Plugins[i].CurrentInstance.ID** : Instance ID de l'instance courante (i = Type ID)
- **Plugins[i].CurrentInstance.Name** : nom de l'instance courante (ou chaîne vide)

Les Plugins

NRBasic

- **Slider Thumb Component** :
- **Image Progress Bar Component** :
- **Text Edit Box Component** :
- **ActiveX Component** :
- **Actions**
 - **Call ActiveX Method** :
 - **Execute command from shell** : exécute une ligne de commande
 - **Exit NetRemote** : quitte le client
 - **Execute lua code (OnDown) / (OnRelease)** : exécute le code quand le bouton est pressé ou relâché
 - **Lua Event** : spécifie le nom d'une fonction Lua existante, ou définit le corps d'une fonction anonyme. Dans les deux cas la fonction prend deux paramètres, le premier est l'objet élément qui correspond au bouton pressé, le second est un nombre : "0" si la fonction est appelée quand le bouton est pressé, si le bouton est maintenu pressé, chaque appel incrémente la valeur, quand le bouton est relâché, la fonction est appelée avec "-1"
 - **Make HTTP request** :
 - **Minimize Application** : minimize le client
 - **Show About** : affiche la boîte "About"
 - **Show Config** : affiche le dialogue de configuration
 - **Hide SIP / Show SIP / Toggle SIP** : sur PPC
 - **Skip the next action if the specified variable does not equal the value** : exécute la prochaine action dans la liste si la variable a la valeur spécifiée
 - **Special Jump** :
 - **Stop all actions if the specified variable does not equal the value** : n'exécute pas la suite des actions de la liste si la variable n'a pas la valeur spécifiée
 - **Toggle device list** : montre/cache une liste de sélection des Page Groups disponibles
 - **Toggle fullscreen** : commute l'écran client entre affichage plein écran ou fenêtre
 - **Toggle window frame** : montre/cache la frame (contour) de la fenêtre client
 - **Turn screen off** : éteint l'écran
 - **Add value to variable** : ajoute une valeur à une variable
 - **Clear image variable** : supprime l'image d'une Variable Image
 - **Set variable to value** : positionne une variable à une valeur spécifiée
 - **Toggle value of variable** : si "0" devient "1", sinon devient "0"
 - **Wait for variable to equal a value** : suspend l'exécution des actions d'une liste jusqu'à ce que la variable vaut la valeur spécifiée ou que le temps soit dépassé

AvidUtils

- **Text String Component** :
- **Scrolling Text Component** :
- **Dropdown Components** :
- **Actions** :
 - **Display text jump down / up** :
 - **Display text scroll down / up** :
 - **Keep awake** : empêche le PPC de s'éteindre tant que l'action est connectée à un Bouton visible sur l'écran et que la variable spécifiée varie. Spécifier **Avid.FlashState** pour un empêchement indéfini, ou une variable de progression d'un lecteur de media par exemple
 - **Play a sound / Play a sound (expanding)** :

- **Set a variable after a delay** : positionne une variable à une valeur donnée après un délai spécifié. Ceci peut être répété après un intervalle donné et pour un nombre de fois spécifié ou indéfini (répétition = 0)
- **Cancel a scheduled set of a variable after a delay** : annule les opérations à venir dues à une action comme ci-dessus
- **Toggle the PPC's SIP** : sur PPC
- **Variables** :
 - **Avid.FlashState** : variable qui change entre "0" et "1" chaque seconde. Util pour faire clignoter un Bouton

FlashPlayer

- **Embedded Macromedia Flash player component** :
- **Actions** :
- **Lua scripting** :

Generic

- **Lua scripting** :
- **GenericPlugin.lua** :

Girder

- **Mouse mode component** : produit un Bouton qui se comporte comme un pad souris virtuel, travaille avec Girder pour piloter la souris du PC avec Girder
 - **Draw Underlying Elements** :
- **Actions** :
 - **Force reconnect to Girder** : ferme la connexion en cours et ouvre une nouvelle
 - **Register variable for feedback** : une fois cette action exécutée, des événements Girder seront générés chaque fois que la variable change de valeur. La chaîne **event string** sera le nom de la variable et le paramètre **P3** de l'événement sera la nouvelle valeur
 - **Send event to Girder** : envoie un événement à Girder avec une chaîne **event string**
- **Variables** :
 - **Girder.LinkActive** : "1" si l'instance par défaut liée à Girder est active, sinon "0"
 - **Girder.n.LinkActive** : comme ci-dessus, mais pour une instance différente de la défaut
 - **Girder.n.FeedbackLineActive** : "1" si le lien de retour de Girder est actif, sinon "0"
 - **Girder.n.GUID** : identifie l'instance en cours de Girder

Infrared

- **Actions** :
 - **Set GlobalCache IR Port Number** :
 - **Start learning mode** :
 - **Infrared** :
- **Variables** :
 - **IR.HostName** :
 - **IR.Macro** :
 - **IR.PortNumber** :

MediaBridge

- **Media Library Tree Component** :
- **Playing Now List Component** :
- **Search Result List Component** :
- **Actions** :
- **Variables** :

Wake On Lan

- **Action :**
 - **Send Wake On Lan :** envoie une commande sur le réseau afin d'allumer le PC contenant la carte réseau avec l'adresse MAC spécifiée

WebBrowser

- **Embedded web browser component :**
- **Properties :**
- **Actions :**
- **Variables :**
- **Web Server Features :**

Zoom

- **Actions :**
- **Variables :**

Les conteneurs de scripts Lua

On peut trouver des scripts Lua dans :

- Les fichiers CCF qui utilisent l'action "Execute Lua Code" du plugin NRBasic : le code est exécuté si le Bouton est pressé, relâché ou maintenu
- Les fichiers CCF comme règle de détermination d'état d'un élément : le code est exécuté si la page est réaffichée pour déterminer l'état à montrer
- Un fichier portant le même nom que le CCF, situé dans le même répertoire, avec l'extension **.lua** : il est chargé et exécuté quand le client charge le fichier CCF. A utiliser pour des fonctions spécifiques au CCF chargé
- Un fichier avec l'extension **.lua** situé dans le sous-répertoire **/luascript/startup/** dans le répertoire de l'exécutable du client : tout fichier de ce type sera chargé et exécuté au démarrage du client, l'ordre de chargement n'étant pas fixé il faut que ces fichiers soient indépendant. A utiliser pour des fonctions disponibles quelque soit le CCF chargé
- Un fichier avec l'extension **.lua** situé dans le même répertoire que le CCF ou dans le sous-répertoire **/luascript/** dans le répertoire de l'exécutable du client, ou dans un répertoire situé dans les répertoires indiqués précédemment : ces fichiers doivent être explicitement chargés à partir d'un autre script avec l'instruction **require**, on peut ainsi gérer une dépendance entre les fichiers
- Inclus dans du code HTML dans un fichier avec l'extension **.html** dans un répertoire **/httpd/** situé dans le répertoire de l'exécutable du client : ce code sera exécuté quand le fichier est servi par le serveur web interne du plugin WebBrowser

Lua Library Reference

Notation

Les noms entre [] sont les paramètres formels, les paramètres de sortie sont optionnels

Une fonction est décrite comme suit :

```
[output1], [output2] = library.function([input1], [input2])
```

où "library" est le nom de la bibliothèque à laquelle appartient la fonction, chaque paramètre est décrit comme suit :

input1: Type. Description

...

On peut utiliser la fonction comme ceci :

```
local v1, v2 = library.function(v3, v4)
v1 = library.function(1.2, "Parsecs")
library.function(1.9 * v3, "Par".."secs")
```

Sauf indication contraire le premier paramètre en sortie vaut **nil** s'il se produit une erreur.

Deux paramètres standards montrés dans la syntaxe mais non décrits :

- **res** : nil si erreur, utilisé si pas de paramètre de sortie requis
- **err** : nombre ou chaîne, information sur l'erreur

Conventions des paramètres formels :

```
[lowercase] -- paramètre normal
[Capitalized] -- paramètre objet
[UPPERCASE] -- nombre ou chaîne pour lequel un ensemble de constantes sont fournies
```

Un paramètre peut être une fonction (*callback*), dans ce cas la syntaxe de la fonction callback est montrée en premier puis la syntaxe de la fonction appelante

```
[output] = [callback]([input1], [input2])
library.function([input3], [callback])
```

Exemple :

```
function mycallback(in1, in2) print(in1, in2); end
library.function("Hello", mycallback)
```

Les objets sont montrés avec la fonction de création en premier, puis les méthodes

```
[MyObj] = library.factory([input])
[MyObj]:Method1([input2])
[ret] = [MyObj]:Method2()
```

Exemple :

```
local obj = library.factory("Hello")
obj:Method1("World")
print(obj:Method2())
```

NetRemote Library

Fonctions de création, d'accès et de contrôle de variables NetRemote

Un observateur (**Watcher**) peut être créé pour chaque variable, permet à des fonctions Lua d'être exécutées

ou à des plugins d'être notifiés si la variable change de valeur.

```
[val] = NetRemote.GetVariable([name])
NetRemote.SetVariable([name], [val])
NetRemote.SetImageVariable([name], [mime], [data])
[Watcher] = NetRemote.RegisterVariableWatch([name], [func], [ifnotnull])
[Watcher] = NetRemote.RegisterVariableWatch([name], [func])
[Watcher] = NetRemote.RegisterVariableWatch([name], [Plugin], [ifnotnull])
[Watcher] = NetRemote.RegisterVariableWatch([name], [Plugin])
NetRemote.UnregisterVariableWatch(Watcher)
Watcher:Unregister()
Watcher:Enable()
Watcher:Disable()
```

- **name** : string. Nom de la variable
- **val** : string. Valeur de la variable, chaîne vide si le nom n'existe pas
- **mime** : string.
- **data** : string.
- **func** : function. fonction lua appelée si la variable change
- **Plugin** : object. objet plugin notifié quand la variable change
- **ifnotnull** : boolean. si true, la fonction est appelée seulement si la variable est positionnée à une nouvelle valeur
- **Watcher** : object. objet variable Watcher qui fournit l'accès aux méthodes qui suivent