Controller - Plugin Development Guide Version 0.4

http://www.crea-doo.at/

31. Dezember 2007

Inhaltsverzeichnis

Einleitung **3**
Das Wichtigste in Kürze . . . 3

Pluginarchitektur **3**
Pluginverwaltung im Controller . . . 3

Entwickeln eigener Plugins **4**
Grundlegende Kenntnisse . . . 4
Commands/Messages . . . 7
Schnittstellen . . . 9
Controller-Objekt . . . 9
Menü-Objekt . . . 10
Menü-Item-Objekt . . . 10

Sonstiges **10**

Unterstützung **11**

Versionshinweise **11**

Anhang **12**
Beispielprojekt - PluginTest.vbp . . . 12
Beispielklasse - cPluginTest.cls . . . 12

Einleitung

Dieses Dokument ist eine Einführung in die Pluginarchitektur für das Programm „Controller“ und basiert auf der Version 1.4.0.7 des „Controller“.

Das Wichtigste in Kürze

Der „Controller“ ist ein Programm zum Steuern vom iTunes und Winamp per SonyEricsson Handy über das Datenkabel, Infrarot oder Bluetooth. Dabei erstellt der „Controller“ - nach erfolgreicher Verbindungsaufnahme mit dem Handy - im „Unterhaltung“-Menü ein neues Untermenü, über das man die Steuerungsaufgaben wahrnehmen kann. Der Titel des Eintrages lautet „iTunes steuern“ oder „Winamp steuern“, je nach eingestelltem Modus.

Pluginarchitektur

Die Plugins, die im Subverzeichnis „plugins“ des „Controller“-Installationsverzeichnisses liegen, werden nachdem das Hauptprogramm geladen hat initialisiert.

Dabei ist zu beachten, dass die Datei „plugins.dat“ das Laden/Entladen der Plugins beeinträchtigen kann. In dieser Datei sind die ID's aller bisher geladener Plugins gespeichert und ihr Zustand darin vermerkt. Diese Datei ist nach dem Format von ini-Dateien aufgebaut und sieht zum Beispiel wie folgt aus:

```
[pLoader.pluginsettings]
Controller.test=Wahr
Controller.Polling=Wahr
```

Falls ein Plugin nicht geladen werden soll hat es das Attribut „Falsch“, „False“ oder „0“ andernfalls „Wahr“, “True“ oder „1“

Die in der Datei gespeicherten Werte können im Fenster „Plugins - Einstellungen“ mittels der Plugins-Auflistung verändert und anschließend gespeichert werden.

Pluginverwaltung im Controller

Im Fenster „Plugins - Einstellungen“ kann man die Zustände der Plugins verändern und wenn sie Konfigurationsmöglichkeiten anbieten, diese Konfiguration vornehmen. Dies geschieht programmintern mit dem UserControl aus der Datei „pLoaderControl.ocx“ die als Komponente ins Projekt eingebunden ist. Für die Plugins bedeutet das, dass diese im laufenden Betrieb über die Funktionen „iPlugin_activate“ und „iPlugin_deactivate“ ein-/ausgeschalten werden können. Diese werden im folgenden Abschnitt noch näher behandelt. Daher muss der Plugin-Entwickler dafür sorgen, dass jegliche Funktionalität die das Plugin bereitstellt beim Aufruf dieser beiden Funktionen bereitgestellt oder entfernt wird. Nach dem Aufruf der „iPlugin_deactivate“ hat das Plugin sonst keine Chance mehr irgendwie in das Geschehen einzugreifen.

Entwickeln eigener Plugins

Um eigene Plugins zu entwickeln, sollte man einigermaßen fundierte Kenntnis der Programmiersprache „Visual Basic 6.0“ besitzen.

Im Anhang (siehe Seite 12) findet sich noch das Listing zum Beispiel-Plugin.

Grundlegende Kenntnisse

Im folgenden werden die einzelnen Schritte, die zum grundsätzlichen Erstellen eines Plugins notwendig sind, kurz erläutert:

- Ein neues Projekt des Typs „ActiveX-DLL“ in Visual Basic 6.0 anlegen.
- Den Verweis zu „pLoader“ im „Verweise“-Dialog setzen
- Den Verweis zu „ControllerHelperLib“ im „Verweise“-Dialog setzen
- In eine beliebige Klasse das Interface „iPlugin“ implementieren

```
Implements iPlugin
```

- Alle Eigenschaften des Interfaces von „Private“ auf „Public“ ändern (!)
- Für alle Eigenschaften Rückgabewerte definieren, wie zum Beispiel:

```
Public Function iPlugin_activate _
    (ByVal eReason As ploader.ePluginActivationReason) _
    As Boolean

    iPlugin_activate = True
End Function

Public Property Get iPlugin_Build() As Long
    iPlugin_Build = 0
End Property

Private Function iPlugin_changeLanguage() As Boolean
    '
End Function

Public Property Get iPlugin_Comments() As String
    iPlugin_Comments = App.Comments
End Property

Public Property Get iPlugin_CompanyName() As String
    iPlugin_CompanyName = App.CompanyName
```

```
End Property

Public Property Get iPlugin_Configurable() As Boolean
    iPlugin_Configurable = False
End Property

Public Function iPlugin_configure() As Boolean
    iPlugin_configure = False
End Function

Public Function iPlugin_deactivate(ByVal eReason As _
    ploader.ePluginDeactivationReason) As Boolean

    iPlugin_deactivate = True
End Function

Public Property Get iPlugin_FileDescription() As String
    iPlugin_FileDescription = App.FileDescription
End Property

Public Property Get iPlugin_id() As String
    iPlugin_id = "controller.testplugin"
End Property

Public Property Get iPlugin_language() As ePluginLanguage
    iPlugin_language = ePluginLanguageDefault
End Property

Public Property Get iPlugin_LegalCopyright() As String
    iPlugin_LegalCopyright = App.LegalCopyright
End Property

Public Property Get iPlugin_LegalTrademarks() As String
    iPlugin_LegalTrademarks = App.LegalTrademarks
End Property

Public Function iPlugin_load(cHost As Object) As Boolean
    iPlugin_load = True
End Function

Public Function iPlugin_loadCallback(cPluginCallback As _
    pLoader2.cPluginCallback) As Boolean

    iPlugin_loadCallback = True
```

```
End Function

Public Property Get iPlugin_Major() As Long
    iPlugin_Major = App.Major
End Property

Public Property Get iPlugin_Minor() As Long
    iPlugin_Minor = App.Minor
End Property

Public Property Get iPlugin_ProductName() As String
    iPlugin_ProductName = App.ProductName
End Property

Public Function iPlugin_receiveCommand _
    (ByVal sCommand As String, _
    ByVal Commands As Collection, _
    Optional ByVal sCallbackCommandId _
    As String = vbNullString) As Variant

    '
End Function

Public Function iPlugin_receiveMessage _
    (ByVal Message As Variant, _
    Optional ByVal sCallbackMessageId _
    As String = vbNullString) As Variant

    '
End Function

Public Property Get iPlugin_Revision() As Long
    iPlugin_Revision = App.Revision
End Property

Public Property Get iPlugin_Title() As String
    iPlugin_Title = App.Title
End Property

Public Function iPlugin_unload _
    (ByVal eReason As ploader.ePluginUnloadReason) _
    As Boolean

    '
```

```
End Function

Private Property Get iPlugin_Url() As String
    iPlugin_Url = "http://www.url.to/plugin/site/"
End Property
```

- Das Plugin als Datei mit der Endung „dll“ kompilieren und die erstellte Datei in das „plugins“-Unterverzeichnis des „Controller“ kopieren.
- Den Controller starten und im Fenster „Plugins - Einstellungen“ prüfen, ob das erstellte Plugin erkannt wird.
- Fertig!

Diese Auflistung der Schritte beschränkt sich natürlich nur auf das Notwendigste.

Die Property „iPlugin_id“ gibt die eindeutige Identifikation eines Plugins fest. Pro Anwendung wird vom pLoader nur ein Plugin mit derselben ID geladen. Mit der ID kann ein Plugin vom Host-Programm auch direkt angesprochen werden. Die ID wird am besten in einer Konstanten gespeichert, sodass sie - da sie im Weiteren an einigen Codestellen gebraucht wird - bei einer Änderung nicht im ganzen Code, sondern zentral geändert werden kann. Für jede ID werden auch diverse Einstellungen gespeichert wie auf Seite 3 beschrieben.

In den beiden Methoden „iPlugin_receiveCommand“ und „iPlugin_receiveMessage“ spielen sich während der Kommunikation mit dem Host-Programm im Weiteren dann die wesentlichsten Dinge ab. An diese beiden Methoden werden dann vom Host-Programm die relevanten Messages und Commands gesendet, auf die die Plugins reagieren können sollen (siehe Abschnitt „Commands/Messages“). Der wesentliche Unterschied zwischen Messages und Commands besteht dabei darin, dass Messages reine Nachrichten vom Type „Variant“ sind, die verändert werden können, und einfach an das nächste Plugin weitergereicht werden. Das heißt das jedes Plugin „seinen Senf“ zu einer Nachricht dazugeben kann. Bei Commands werden spezielle Commands an die Plugins geschickt, die durch die Variable „sCommand“ eindeutig identifiziert werden können und zusätzlich in der Collection „Commands“ noch beliebig viele weitere Daten mitführen können. Die Menge dieser Daten hängt dabei einzig und allein vom Host-Programm ab!

Die grundlegende Kommunikation läuft nur über die oben beschriebenen Methoden ab. Darüberhinaus besteht die Möglichkeit über das „cHost“-Objekt und das „cPluginCallback“-Objekt mit dem Host-Programm in Verbindung zu treten. Dabei legt das „cHost“-Objekt die Schnittstelle zu den vom Host-Programm angebotenen Methoden frei und das „cPluginCallback“-Objekt ermöglicht das Senden von Messages und Commands und das permanente Speichern von beliebigen Einstellungen. Diese gesetzten Einstellungen werden dann vom Host-Programm in einer separaten Datei gespeichert. Im Falle das Controllers ist dies wiederum die Datei „plugins.dat“ im Unterverzeichnis „plugins“.

Commands/Messages

Die im folgenden beschriebenen Commands/Messages werden vom Controller verwendet:

„Controller“ ab Version 1.4.0.7	Beschreibung
Command	
controller.activated	Inhalt der Commands-Collection: „MAIN“ - Controller-Objekt
controller.loaded	Inhalt der Commands-Collection: „MAIN“ - Controller-Objekt
controller.menu	Inhalt der Commands-Collection: „id“ - Menü-ID „item“ - Menü-Item-Objekt
controller.menu_advanced_create	Inhalt der Commands-Collection: „menu“ - Menü-Objekt
controller.menu_debug_create	(siehe controller.menu_advanced_create)
controller.menu_ituneswatcher_create	(siehe controller.menu_advanced_create)
controller.menu_main_create	(siehe controller.menu_advanced_create)
controller.menu_other_create	(siehe controller.menu_advanced_create)
controller.menu_plugins_create	(siehe controller.menu_advanced_create)
controller.menu_volume_create	(siehe controller.menu_advanced_create)
Message	
controller.connected	Verbindung zum Handy hergestllt
controller.connecting	Verbindung zum Handy hergestellen
controller.connectionerror	Verbindung zum Handy fehlgeschlagen
controller.disconnected	Verbindung zum Handy getrennt
controller.disconnecting	Verbindung zum Handy wird getrennt
controller.loaded	„Controller“ ist fertiggeladen
controller.menu_inmenu_false	Handy befindet sich im Menü
controller.menu_inmenu_true	Handy befindet sich nicht im Menü
controller.menu_shutdown	„Controller“ wird per Handy beendet
controller.menu_shutdown_app	iTunes/Winamp wird per Handy beendet
controller.menu_shutdown_pc	Der PC wird per Handy beendet
controller.settings_display	Dialog „Einstellungen“ wird geladen
controller.settings_displayed	Dialog „Einstellungen“ wurde geladen
controller.settings_hide	Dialog „Einstellungen“ wird geschlossen
controller.settings_hided	Dialog „Einstellungen“ wurde geschlossen
controller.unloading	„Controller“ wird beendet

Schnittstellen

Im folgenden Abschnitt werden die im vorigen Abschnitt zum Teil verwendeten Objekte näher erläutert.

Controller-Objekt

Name	Return	Type	Parameter
CurrentApplication	Long	Property Get	-
CurrentPort	Long	Property Get	-
isComPortAvailable	Boolean	Property Get	ByVal lPortNumber As Long
isConnected	Boolean	Property Get	-
isValidManufacturer	Boolean	Property Get	-
isValidModel	Boolean	Property Get	-
iTunesExist	Boolean	Property Get	-
iTunesWatcherState	Boolean	Property Get	-
Manufacturer	String	Property Get	-
Model	eModel	Property Get	-
ModelCode	String	Property Get	-
ModelSeries	eModelSeries	Property Get	-
ModelSeriesString	String	Property Get	-
ModelString	String	Property Get	-
performDisConnect	Boolean	Function	-
performWinampSearch	Boolean	Property Get	-
Send2Phone	PhoneCallback	Function	ByVal sATString As String
SettingsPath	String	Property Get	-
showMessage	-	Sub	Text As String, Optional Timeout As Integer
StayInSearchMenu	Boolean	Property Get	-
TickerState	Boolean	Property Get	-
uploadCurrentTrack	Boolean	Function	bMemoryStick As Boolean
uploadFile2Phone	Boolean	Function	ByVal sFile As String, ByVal sFilename As String, ByVal sFolder As String, bMemoryStick As Boolean
WinampExist	Boolean	Property Get	-

Menü-Objekt

Name	Return	Type	Parameter
addItem	Menu-Item	Function	sCaption As String Optional sTag As String Optional sTag2 As String Optional iIcon As Integer = 0 Optional bDimmed As Boolean = False Optional bDeletable As Boolean = False
Count	Long	Property Get	-
Item	Menu-Item	Property Get	-
ItemFromId	Menu-Item	Property Get	-
Tag	String	Property Get	-
TagFromId	String	Property Get	-

Menü-Item-Objekt

Name	Return	Type	Parameter
Caption	String	Property Get/Let	-
Deletable	Boolean	Property Get/Let	-
Dimmed	Boolean	Property Get/Let	-
Icon	Integer	Property Get/Let	-
Id	String	Property Get/Let	-
Selected	Boolean	Property Get/Let	-
Tag	String	Property Get/Let	-
Tag2	String	Property Get/Let	-

Sonstiges

Wenn Sie näheres zum Programm „Controller“ erfahren möchten besuchen Sie bitte die folgenden Links:

- http://www.crea-doo.at/weblog/2006/09/07/ituneswinamp-controller/
- http://development.crea-doo.at/vb/controller/
- http://www.fjsoft.at/forum/viewtopic.php?t=839
- http://www.se-foren.de/thread.php?threadid=16795
- http://www.se-world.info/thread158271-Winamp-Informationen-ueber-Bluetooth.html

Unterstützung

Der „Controller“ ist Freeware - die Softwareentwicklung kostet jedoch viel Zeit und Geld. Falls Ihnen das Programm gefällt, tragen Sie bitte mit Ihrer Spende auch einen Teil dazu bei, dass ich den „Controller“ auch in Zukunft weiterentwickeln und kostenfrei zur Verfügung stellen kann. Um zu Spenden müssen Sie lediglich auf folgenden Link klicken:

- Spenden mit Paypal

Versionshinweise

- Version 0.1: Erste Veröffentlichung
- Version 0.2: Kleinere Fehler ausgebessert, die Abschnitte „Commands/Messages“ & „Schnittstellen“ hinzugefügt und an den „Controller“ in der Version 1.3.0.11 angepasst.
- Version 0.3: Kleinere Fehler ausgebessert, die Abschnitte „Commands/Messages“ & „Schnittstellen“ an den „Controller“ in der Version 1.3.0.43 angepasst.
- Version 0.4: Kleinere Fehler ausgebessert, die Abschnitte „Commands/Messages“ & „Schnittstellen“ an den „Controller“ in der Version 1.4.0.7 angepasst.

Anhang

Beispielprojekt - PluginTest.vbp

```
1  Type=OleDll
2  Reference=*\G{00020430-0000-0000-C000-000000000046}#2.0#0#C:\
       Windows\system32\stdole2.tlb#OLE Automation
3  Reference=*\G{26A75036-F521-4EA7-BAC4-A34EB7408FBA}#7a.0#0#
       pLoader2.dll#pLoader2
4  Reference=*\G{AD91B3E1-3DB4-432E-9FDE-10AE1DF04518}#4.0#0#
       ControllerHelperLib.dll#ControllerHelperLib
5  Class=cPluginTest; cPluginTest.cls
6  Startup="Sub Main"
7  HelpFile=""
8  Title="PluginTest"
9  Command32=""
10 Name="PluginTest"
11 HelpContextID="0"
12 Description="PluginTest"
13 CompatibleMode="1"
14 MajorVer=1
15 MinorVer=0
16 RevisionVer=0
17 AutoIncrementVer=0
18 ServerSupportFiles=0
19 CompilationType=0
20 OptimizationType=0
21 FavorPentiumPro(tm)=0
22 CodeViewDebugInfo=0
23 NoAliasing=0
24 BoundsCheck=0
25 OverflowCheck=0
26 FlPointCheck=0
27 FDIVCheck=0
28 UnroundedFP=0
29 StartMode=1
30 Unattended=0
31 Retained=0
32 ThreadPerObject=0
33 MaxNumberOfThreads=1
```

Beispielklasse - cPluginTest.cls

```
1  Option Explicit
2  '------------------------------------------------------------
```

```
3  Implements iPlugin
4  '------------------------------------------------------------
5  Private Const sPluginId As String = "Controller.PluginTest"
6  '------------------------------------------------------------
7  Private COM As Object
8  '------------------------------------------------------------
9
10 Public Function iPlugin_activate(ByVal eReason As pLoader2.
       ePluginActivationReason) As Boolean
11     On Local Error GoTo errError
12
13     iPlugin_activate = True
14
15     On Error GoTo 0
16     Exit Function
17
18 errError:
19     MsgBox "Error (" & Err.Number & "):" & Err.Description & "
           Klassenmodul cPluginTest; procedure: iPlugin_activate"
20     Resume Next
21 End Function
22
23 Public Property Get iPlugin_Build() As Long
24     iPlugin_Build = 0
25 End Property
26
27 Public Function iPlugin_changeLanguage() As Boolean
28     iPlugin_changeLanguage = True
29 End Function
30
31 Public Property Get iPlugin_Comments() As String
32     iPlugin_Comments = App.Comments
33 End Property
34
35 Public Property Get iPlugin_CompanyName() As String
36     iPlugin_CompanyName = App.CompanyName
37 End Property
38
39 Public Property Get iPlugin_Configurable() As Boolean
40     iPlugin_Configurable = False
41 End Property
42
43 Public Function iPlugin_configure() As Boolean
44     iPlugin_configure = False
```

```
45 End Function
46
47 Public Function iPlugin_deactivate(ByVal eReason As pLoader2.
      ePluginDeactivationReason) As Boolean
48     On Local Error GoTo errError
49
50     iPlugin_deactivate = True
51
52     On Error GoTo 0
53     Exit Function
54
55 errError:
56     MsgBox "Error (" & Err.Number & "):" & Err.Description & "
          Klassenmodul cPluginTest; procedure: iPlugin_deactivate"
57     Resume Next
58 End Function
59
60 Public Property Get iPlugin_FileDescription() As String
61     iPlugin_FileDescription = App.FileDescription
62 End Property
63
64 Public Property Get iPlugin_id() As String
65     iPlugin_id = sPluginId
66 End Property
67
68 Public Property Get iPlugin_language() As ePluginLanguage
69     iPlugin_language = ePluginLanguageDefault
70 End Property
71
72 Public Property Get iPlugin_LegalCopyright() As String
73     iPlugin_LegalCopyright = App.LegalCopyright
74 End Property
75
76 Public Property Get iPlugin_LegalTrademarks() As String
77     iPlugin_LegalTrademarks = App.LegalTrademarks
78 End Property
79
80 Public Function iPlugin_load(cHost As cHost) As Boolean
81     Set Host = cHost
82
83     iPlugin_load = True
84 End Function
85
```

```
86  Public Function iPlugin_loadCallback(cPluginCallback As
        cPluginCallback) As Boolean
87      Set PluginCallback = cPluginCallback
88
89      iPlugin_loadCallback = True
90  End Function
91
92  Public Property Get iPlugin_Major() As Long
93      iPlugin_Major = App.Major
94  End Property
95
96  Public Property Get iPlugin_Minor() As Long
97      iPlugin_Minor = App.Minor
98  End Property
99
100 Public Property Get iPlugin_ProductName() As String
101     iPlugin_ProductName = App.ProductName
102 End Property
103
104 Public Function iPlugin_receiveCommand(ByVal sCommand As String,
        ByVal Commands As Collection, Optional ByVal
        sCallbackCommandId As String = vbNullString) As Variant
105 Dim colTemp As New Collection
106
107     On Local Error GoTo errError
108
109     MsgBox sCommand
110
111     Select Case sCommand
112     Case "controller.activated"
113         If Not Commands("COM") Is Nothing Then
114             Set COM = Commands("COM")
115         End If
116     Case Else
117         '
118     End Select
119
120     On Error GoTo 0
121     Exit Function
122
123 errError:
124     MsgBox "Error (" & Err.Number & "):" & Err.Description & "
            Klassenmodul cPluginTest; procedure:
            iPlugin_receiveCommand"
```

```
125     Resume Next
126 End Function
127
128 Public Function iPlugin_receiveMessage(ByVal Message As Variant,
        Optional ByVal sCallbackMessageId As String = vbNullString)
    As Variant
129     MsgBox CStr(Message)
130 End Function
131
132 Public Property Get iPlugin_Revision() As Long
133     iPlugin_Revision = App.Revision
134 End Property
135
136 Public Property Get iPlugin_Title() As String
137     iPlugin_Title = App.Title
138 End Property
139
140 Public Function iPlugin_unload(ByVal eReason As pLoader2.
    ePluginUnloadReason) As Boolean
141     Select Case eReason
142     Case ePluginUnloadReasonDefault
143         '
144     Case ePluginUnloadReasonTermination
145         '
146     End Select
147
148     iPlugin_unload = True
149 End Function
150
151 Public Property Get iPlugin_Url() As String
152     iPlugin_Url = ""
153 End Property
```