Media Management and Automation System for Broadcast

digispot® System GmbH

TRACT

Plug-ins

USER MANUAL

May 2015

TABLE OF CONTENTS

Introduction

Plug-ins for Digispot II system can be of two basic types — window and non-window ones. Window plug-ins operate as conventional Digispot II modules. Non-window plug-ins have no windows and simply exist in the system.

Both plug-in types interact with the application and, possibly, other plug-ins via **Digispot Managed API**.

Creation in VID file

Window Plug-ins

To create a window plug-in you need to insert the following line in the **VID** file:

```
CREATE_NET_WND_PLUGIN ID FILE TYPE
```

Where

- **ID** - is object identifier in Digispot II system
- **FILE** - is the path to the file in which the plug-in is realized. If the path is relative, it is read from the directory where the launchable application is stored
- **TYPE** - is the type (class) of the newly created object. It must be a public type having public constructor with no parameters and derived from the following interfaces: IBasePlugin, IWindowPlugin. The type is specified completely, indicating namespaces .

For example:

```
CREATE_NET_WND_PLUGIN      PlanPlug      ManagedPlugins.dll
Digispot.ManagedPlugins.Plan.PlansEditor
```

Subsequently the newly created object must be placed on a certain area of graphic interface (view), similar to conventional modules — database, schedule editor, player, etc.

Note: *the specified ID is the identifier of the service module that creates the plug-in. If you need to refer to the plug-in directly when addressing events, you must add "-Plugin" to the identifier.*

In the example shown above the plug-in address in the system of events is: **!PlanPlug-Plugin**.

Non-Window Plug-ins

A non-window plug-in does not have its own graphic interface. To create a non-window plug-in, you need to insert the following line in the **VID** file:

```
CREATE _ NET _ PLUGIN ID FILE TYPE
```

Where

- **ID** - is object identifier in Digispot II system
- **FILE** - is the path to the file in which the plug-in is realized. If the path is relative, it is read from the directory where the launchable application is stored
- **TYPE** - is the type (class) of the newly created object. It must be a public type having public constructor with no parameters and derived from the following interfaces: IBasePlugin. The type is specified completely, indicating namespaces.

For example:

```
CREATE _ NET _ PLUGIN kra _ disp managedplugins.dll Kra.Dispatcher
```

This line should be enough to create an object.
In contrast to the window object:

- no extra lines are needed
- specified ID is the direct identifier of plug-in.

Additional Parameters in VID File

Depending on the type of created object, it may need additional parameters specific for this type of plug-in to be specified to operate correctly. (For this, the plug-in must implement the IConfigMember interface).
To set additional parameters in the file you need to insert a line (lines) after the plug-in creation line that should look like this:

```
SET _ OBJ _ BASE ID param _ 1 param _ 2 param _ 3 ... param _ i
```

where:

- **ID** - is identifier, specified during creation of plug-in (no need to add -Plugin)
- **param_1 ... param_i** - are a set of parameter lines separated by spaces. The meaning and the number of parameters are defined by the plug-in and must be specified in its description. If you need to pass one line containing blank space as one parameter, the former must be enclosed in double quotes.

For example:

```
CSET _ OBJ _ BASE kra _ disp WholeScale 40
SET _ OBJ _ BASE kra _ disp Labels «12, 6, 3, 0, -3, -6, -12, -20»
```

Dynamic Compilation

Plug-ins can be created not only from finished **.NET assemblies**, but also from original texts of programs.
When doing this, the following conditions must be observed

- the file of the original text of program must be in C# language (it must have the .cs extension) or VB.NET (the .vb extension)
- the files are compiled with .NET 3.5 version
- during compilation, the following is specified as references:
  - System.dll
  - System.Windows.Forms.dll
  - cmn[d].dll
  - the executed application
- if you need to activate additional references you must insert a line with comment at the beginning of the program text (// for C# and ‘ for VB) that looks like this // InlineReference:System.Core.dll

In this particular case the **System.Core.dll** additional library is being activated.

Debug Log

The following lines are inserted into debug log, to MAIN thread, when creating plug-ins:

- After successful compilation (only for original texts):

```
 @  MAIN  @  Plugins  @  D:\s\#CURRENT\EXE\SYSTEM\ind _ test.cs
successfully  compiled.  References:System.dll,System.Windows.
Forms.dll,D:\s\#CURRENT\EXE\Cmnd.dll,D:\s\#CURRENT\EXE\Cmnd.dll @
Main
```

- After succesful loading of an assembly:

```
 MAIN @ Plugins @ Assembly successfully loaded:D:\s\#CURRENT\EXE\
ManagedPlugins.dll @ Main
```

- After succesful creation of plug-in:

```
 @ MAIN @ Plugins @ Object created. File:D:\s\#CURRENT\EXE\SYSTEM\
ind _ test.cs, class:TractPlugin.Windowless @ Main
```

- If errors occur during creation of a plug-in, these will be displayed in standard error window and additionally saved in debug log.

For example:

```
 @ ERR _ MSG @ ASSERT @ .\ManagedPlugin.cpp Line:791 ProcessConfig
returned false:mirror LeftFlash LeftFlash2  @ Main
```

or

```
 @ ERR _ MSG @ ASSERT @ .\ManagedPlugin.cpp Line:797 Falied to
call ProcessConfig.
Str:FallMult 0.25
Exception:System.FormatException: Input string was not in a
correct format.
     at System.Number.StringToNumber(String str, NumberStyles
options, NumberBuffer& number, NumberFormatInfo info, Boolean
parseDecimal)
```

```
     at  System.Number.ParseSingle(String  value,  NumberStyles
options, NumberFormatInfo numfmt)
      at  System.Single.Parse(String  s,  NumberStyles  style,
NumberFormatInfo info)
   at System.Convert.ToSingle(String value)
     at  Kra.Dispatcher.ProcessConfig(String[]  strings)  in  D:\
s\#CURRENT\COMPONENTS\ManagedPlugins\KRA\KraDispatcher.cs:line
323
  at NativeWindowlessPlugin.ProcessConfig(NativeWindowlessPlugin*
, CStringT<char\,StrTraitMFC _ DLL<char\,ATL::ChTraitsCRT<char> >
>** str _ arr, Int32 cnt) @ Main
```

Apart from these records, the plug-in can make its own records in debug log all by itself.

Example

An example can be found on this page.

Contact information

DIGISPOT System GmbH
Grillparzerstraße 6a D - 22085, Hamburg, Germany
Tel.: +49 (40) 229-88-83, Fax: +49 (40) 22-32-09

http://www.digispot.com
e-mail: support@digispot.com

TRACT-SOFT, LLC
197101, ul. Kronverkskaya, 23, Saint-Petersburg, Russia
Tel.: +7 (812) 490-77-99, Fax: +7 (812) 233-61-47

http://www.digispot.ru
e-mail: support@digispot.ru