Event Service API manual

Table of Contents

Event Service API

Cinegy Air has the possibility to add the controlling event to any item which will be sent to the event service. This service will process events in the specified way. Thus the system flexibility and scalability can be achieved and for each particular case the specified handler can be created.

The event itself is a text command with 5 parameters:

Parameter	Type	Description
Device	String	Unique device name
Command	String	Device command
Parameter 1	String	Context parameter 1
Parameter 2	String	Context parameter 2
Parameter 3	String	Context parameter 3

It's not recommended to use the following names for devices, they are reserved for the system use:

- LOGO;

- RTP_IN.

Technically the event handler is an independent service which implements COM object with the following interface.

IPlayOutExEventSink Interface

Interface provides method for events processing.

Class Hierarchy

IDispatch
IPlayOutExEventSink

Syntax

```
[ object, uuid(646BD034-19BD-415A-A47B-5F12C790062B), dual, nonextensible,
pointer_default(unique) ]
interface IPlayOutExEventSink : IDispatch;
```

Methods

	Name	Description
◆	OnEvent	Method handles the event.

OnEvent

Method handles the event.

Syntax

```
[ id(1) ]
HRESULT OnEvent(
    [in] BSTR bstrDevice,
    [in] BSTR bstrCommand,
    [in] BSTR bstrOp1,
    [in] BSTR bstrOp2,
    [in] BSTR bstrOp3
);
```

Parameters

bstrDevice

specifies the device name.

bstrCommand

specifies the device command.

bstrOp1

specifies the context parameter 1.

bstrOp2

specifies the context parameter 2.

bstrOp3

specifies the context parameter 3.

Returns

Returns S_OK if successful or an error value otherwise.

Events Service

The events system was designed to provide the maximal flexibility for the external event handler developers for handling their devices.

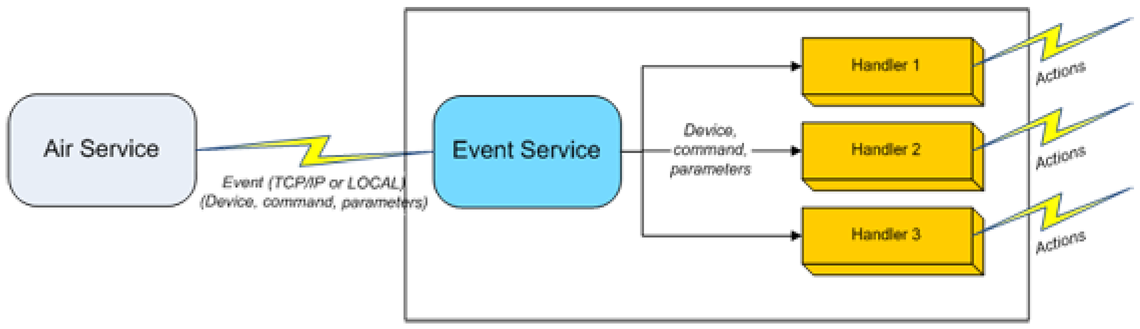
If it is necessary to support several devices then the specified Event Service should be created to receive the events and internally process different commands and send them to different devices. In doing so the Events Service can handle all events internally or manage the internal system of plug-ins.

For each playout server there can be only one Event Service in the system (it is explicitly set up in the server configuration). That's why the Event Server should handle all the commands. If it is necessary to create several modules to handle different commands independently then this structure should be created inside the events handler. The described method is the simplest but it implies the creation of a completely stand-alone service for all needs.

If the collateral use of the external event handlers along with developed by Cinegy is required then another model of the event handler implementation should be used. It takes only to implement the handler plug-ins for each device and the existent Cinegy developed Event Service can be used.

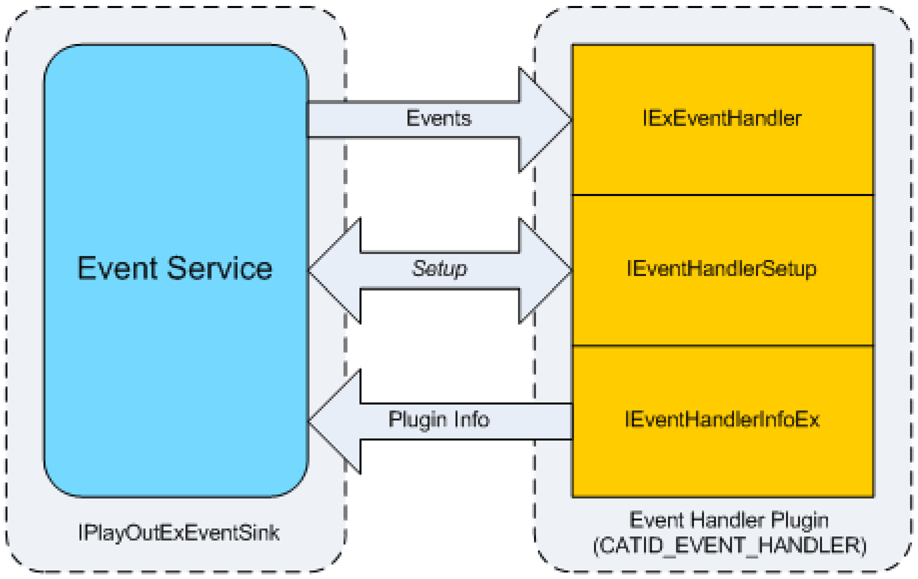
Plug-in implementation

The general chart of the events handling that Cinegy uses can be represented by the following diagram:



There is an "Event Service" which receives all the commands from Air Service and then broadcasts them to all registered plug-ins “Handler 1”, “Handler 2” and so on. The handler itself is a COM object that implements the specifies set of interfaces (see below).

The schematic interaction of the Event Service with each handler can be represented as follows:



Apparently, each plug-in has to meet the following demands:

- to implement IExEventHandler, IEventHandlerSetup and IEventHandlerInfoEx interfaces;

- to be registered in the CATID_EVENT_HANDLER COM-category.

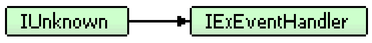
If the plug-in has no own settings it is not necessary to implement IEventHandlerSetup, the other two are mandatory.

The Event Service itself implements IplayOutExEventSink interface and receives events from the Playout service directly.

IExEventHandler Interface

Base interface for plug-in initialization and events handling. This interface should be implemented by each handler.

Class Hierarchy



Syntax

```
[ object, uuid(BF89009C-AFC2-4e0d-BF5C-2FD4602C03E1), pointer_default(unique) ]
interface IExEventHandler : IUnknown;
```

Methods

	Name	Description
	Init	Method initializes the plug-in.
	SendEvent	Method handles the event.
	ShutDown	Method shuts down the plug-in.

Init

Method initializes the plug-in.

Syntax

```
HRESULT Init(
    [in] IUnknown* piEventLog
);
```

Parameters

piEventLog

pointer to the object implementing the IEventLog interface for outputting information to the Log window (e.g. the service console window).

Returns

Returns S_OK if successful or an error value otherwise.

SendEvent

Method handles the event.

Syntax

```
HRESULT SendEvent(
    [in] BSTR bstrDevice,
    [in] BSTR bstrCommand,
    [in] BSTR bstrOp1,
    [in] BSTR bstrOp2,
    [in] BSTR bstrOp3
);
```

Parameters

bstrDevice

specifies the device name.

bstrCommand

specifies the device command.

bstrOp1

specifies the context parameter 1.

bstrOp2

specifies the context parameter 2.

bstrOp3

specifies the context parameter 3.

Returns

Returns S_OK if successful or S_FALSE if event wasn't recognized.

ShutDown

Method shuts down the plug-in.

Syntax

```
HRESULT ShutDown();
```

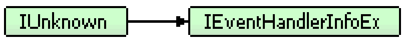
Returns

Returns S_OK if successful or an error value otherwise.

IEventHandlerInfoEx Interface

Interface provides informational methods for passing the information about plug-in to the host-service.

Class Hierarchy



Syntax

```
[ object, uuid(535358E4-F1D1-4a58-9960-5B99782F16D3), pointer_default(unique) ]
interface IEventHandlerInfoEx : IUnknown;
```

Methods

	Name	Description
◆	GetInfo	Method retrieves the plug-in information.
◆	GetPluginName	Method retrieves the plug-in name (to be displayed in the service setup utility).

GetInfo

Method retrieves the plug-in information.

Syntax

```
HRESULT GetInfo(
    [out] BSTR* pbstrDeviceInfo
);
```

Parameters

pbstrDeviceInfo

pointer to the variable that receives the information as a string.

Returns

Returns S_OK if successful or an error value otherwise.

GetPluginName

Method retrieves the plug-in name (to be displayed in the service setup utility).

Syntax

```
HRESULT GetPluginName(
    [out] BSTR* pbstrName
);
```

Parameters

pbstrName

pointer to variable that receives the plug-in name.

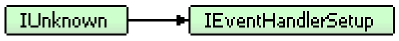
Returns

Returns S_OK if successful or an error value otherwise.

IEventHandlerSetup Interface

Interface provides method for the plug-in setup.

Class Hierarchy



Syntax

```
[ object, uuid(E3BF96C9-F0F9-4bad-B6B0-D72F4D5F75C2), pointer_default(unique) ]
```

```
interface IEventHandlerSetup : IUnknown;
```

Methods

	Name	Description
	Setup	Method sets up the Event Handler.

Setup

Method sets up the Event Handler.

Syntax

```
HRESULT Setup(
    [in] HWND hwndParent
);
```

Parameters

hwndParent

handle to the dialog window with the setup parameters.

Returns

Returns S_OK if successful or an error value otherwise.

Event service configuration

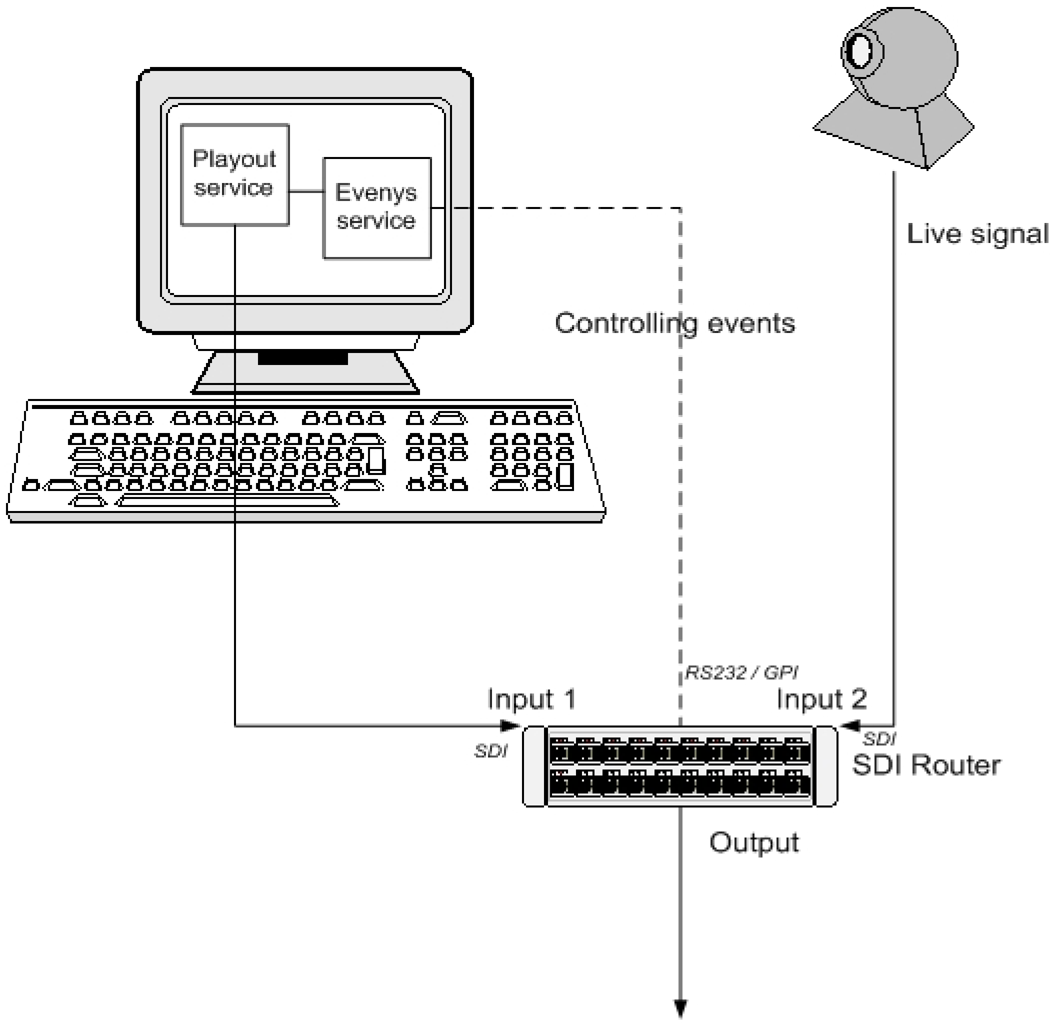
In order to support external devices (e.g. external SDI switches) Cinegy Air application supports the secondary events, which can be used to control the external devices.

There are actually two ways to configure event service: locally and remotely. The local event service is installed on the playout server and can be used in case you have e.g. just one channel which should be switched between playout and external live signal through external SDI switcher.

Remote event service allows you to control one video switcher from the several playout servers (channels) independently. In this case you have to install the event service on the separate PC which should be connected to the controlled device (see detailed description below).

Local Event Service installation

The diagram below shows the typical configuration, if you need to control just one device for one playout server.



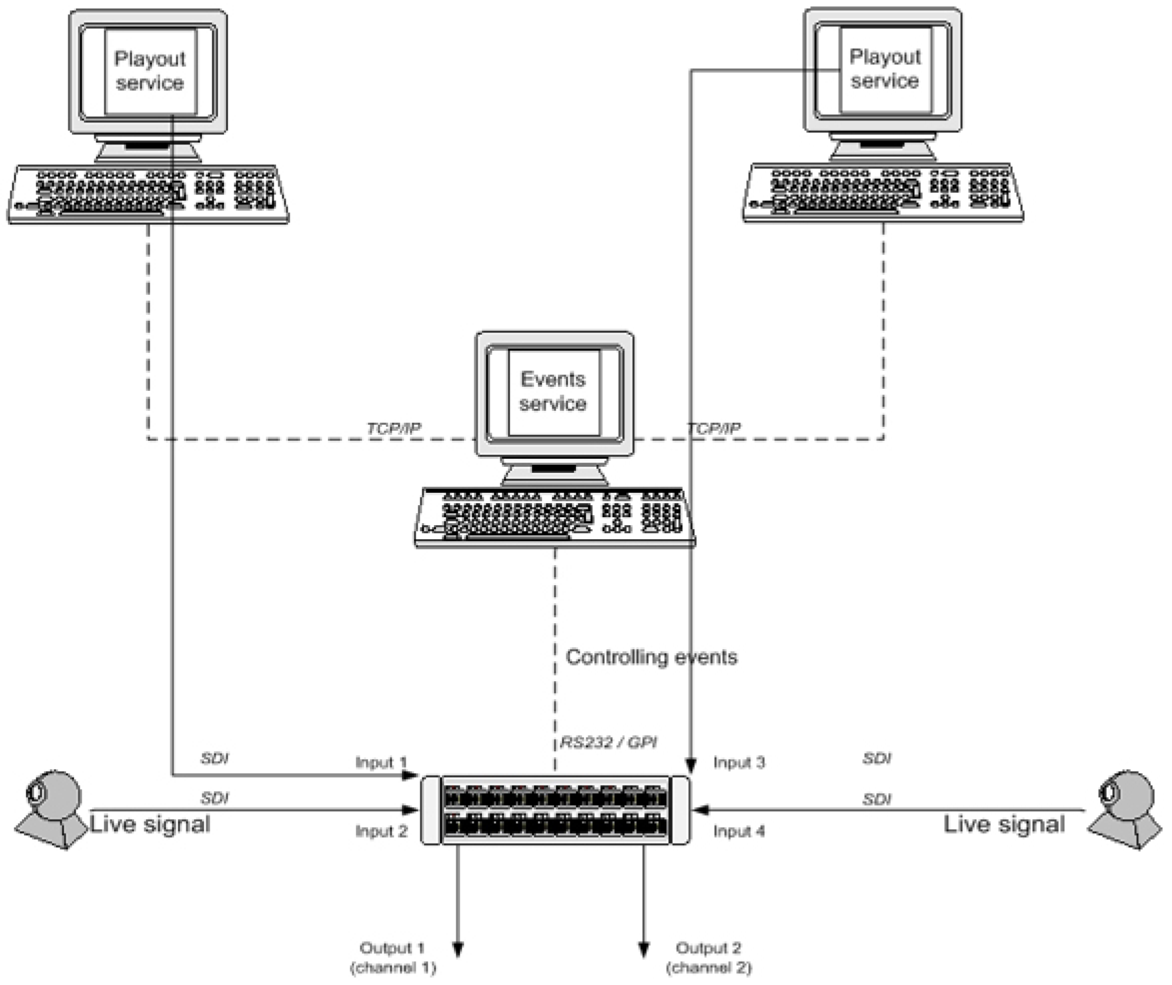
This case shows the situation, when you have one playout server and one external live signal, which should be switched by means of the secondary events (via local event service).

To install external event service locally you have to copy the SDIRouterManager folder to the playout server and then run InsatallLocalService.bat from this folder. After that the corresponding components are installed and registered. In this case video switcher should be connected to this PC via RS232 port (COM).

Remote Event Service installation

If you need to control one device from different playout servers simultaneously, you have to install event service on the separate PC, which should be connected to this device. In this case you have to set up remote access to this PC (via network) from each playout server, which should control the external device.

The diagram below shows one of the possible configurations with remote event service.

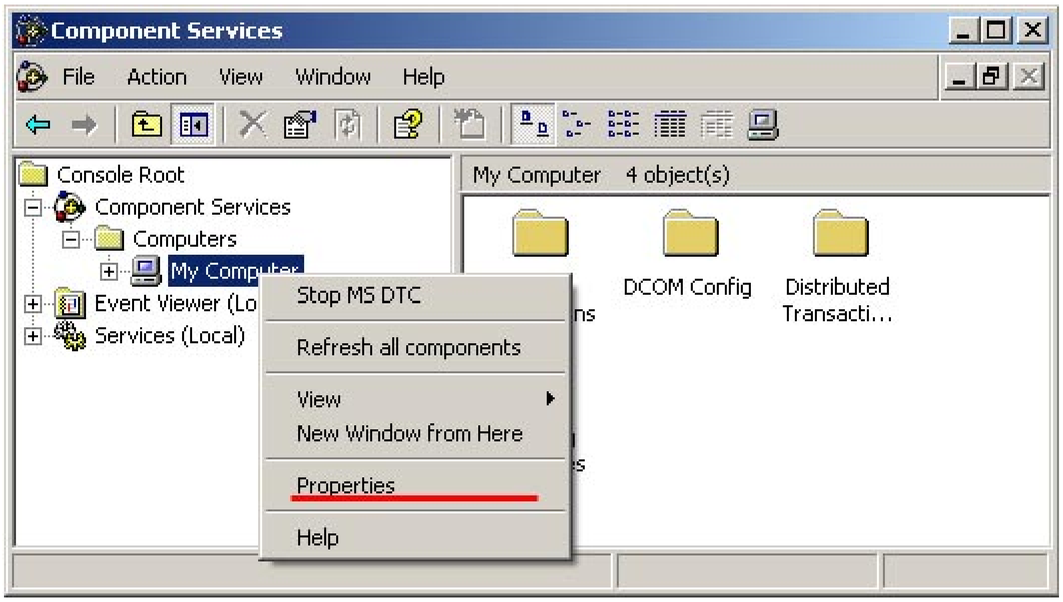


As you can see, this configuration allows using one external device for different channels independently.

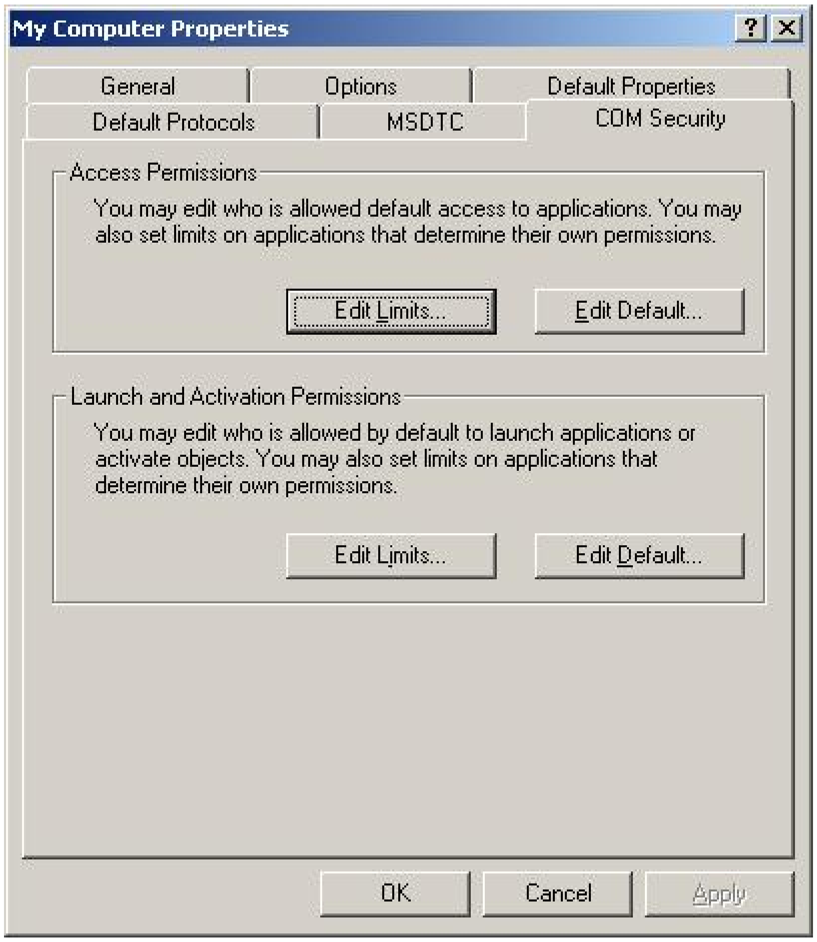
The installation is executed in several steps.

On the remote PC

To install external event service remotely, you have to copy the SDIRouterManager folder to the separate PC, which should be connected to the video switcher and then run InsatallRemoteService.bat from this folder. Then you have to set all the remote permissions to make possible to start the service remotely. Please open “Administrative Tools” folder in Control Panel and double click to “Components Services”. Then right click to “My Computer” in the appeared window and select “Property” command from the popup menu:



The My Computer Properties Dialog appears. Please select the “COM Security “tab as sown below:



My Computer Properties

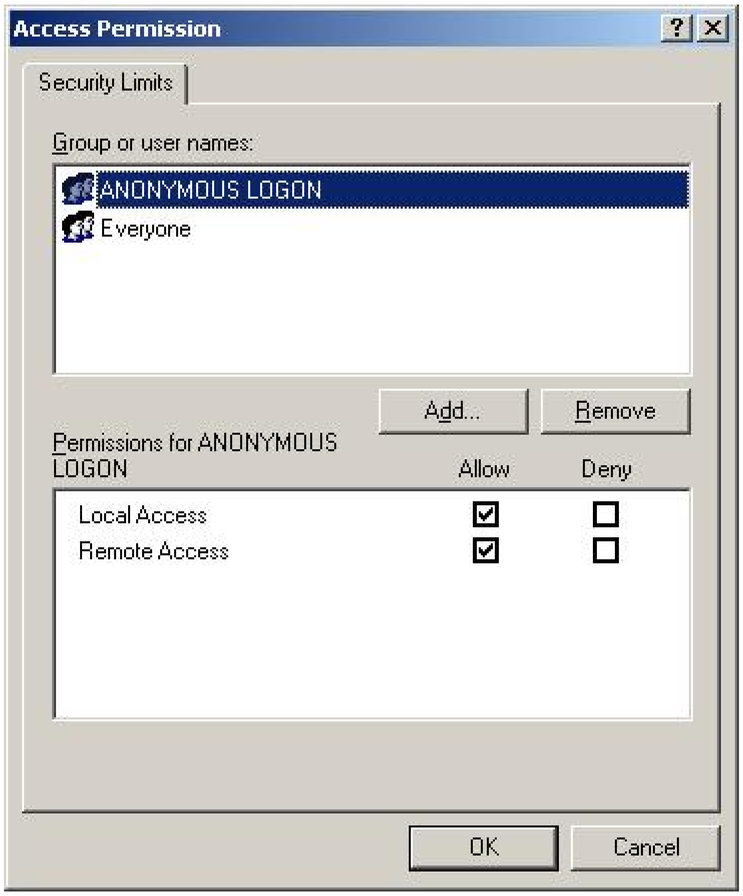
Access Permissions

You may edit who is allowed default access to applications. You may also set limits on applications that determine their own permissions.

Launch and Activation Permissions

You may edit who is allowed by default to launch applications or activate objects. You may also set limits on applications that determine their own permissions.

Then press the “Edit Limits” buttons for the both section and allow all the remote rights for the Everyone user (or for the appointed user, which is logged on on all the playout servers) as shown:



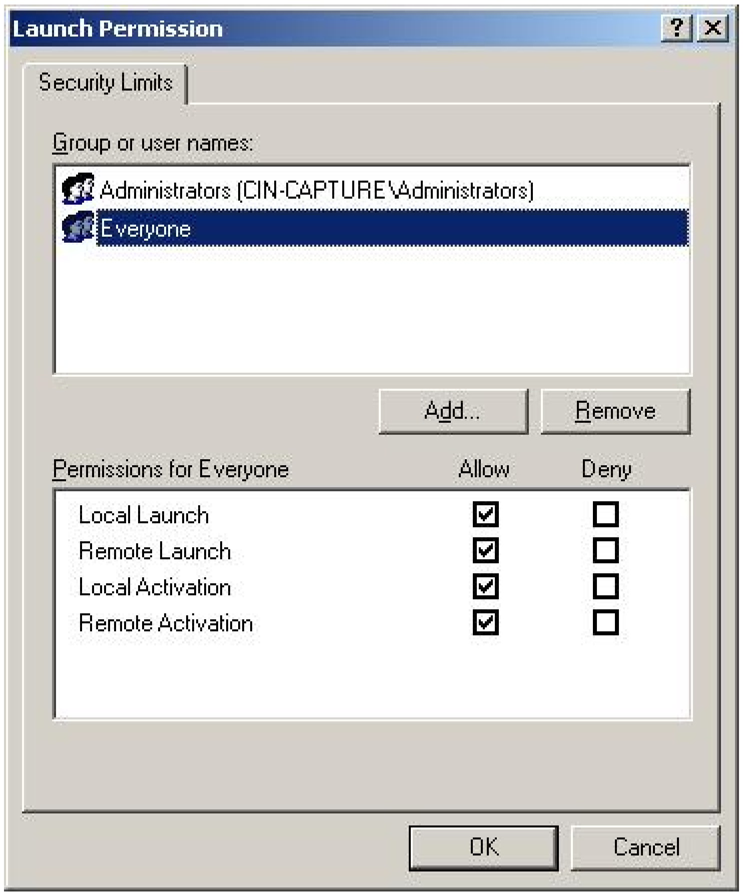
Access Permission

Security Limits

Group or user names:

ANONYMOUS LOGON
Everyone

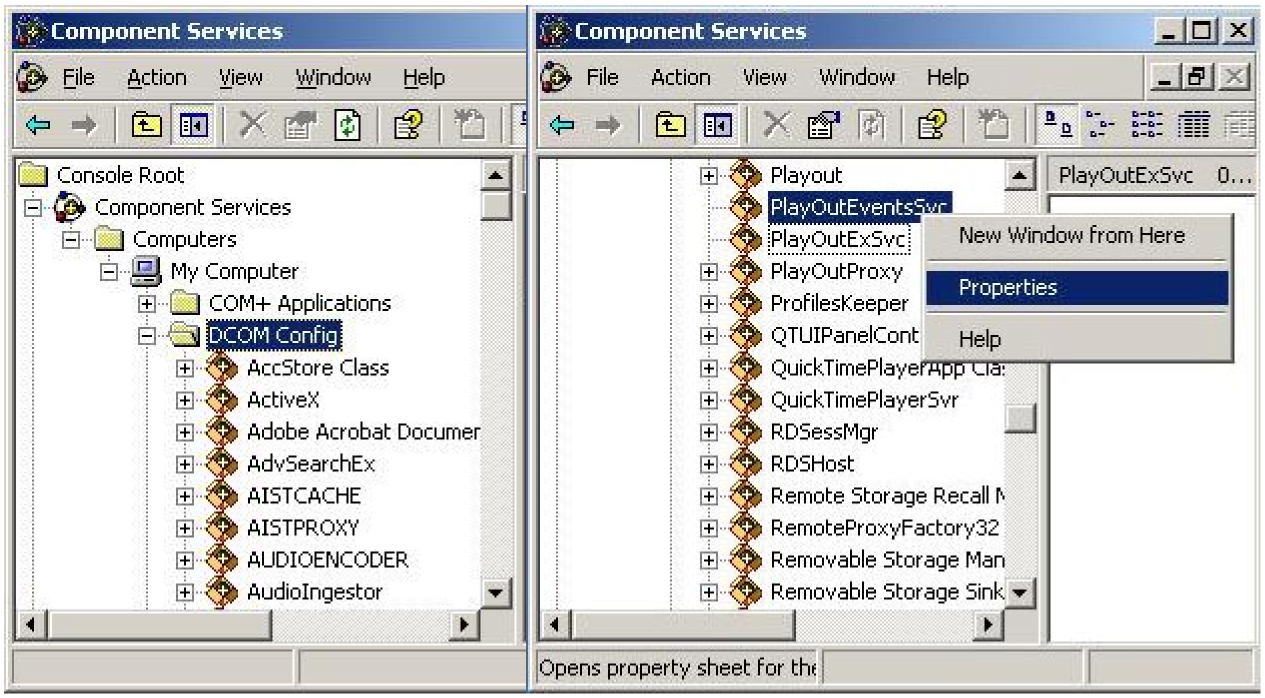
Permissions for ANONYMOUS LOGON	Allow	Deny
Local Access	✓	☐
Remote Access	✓	☐



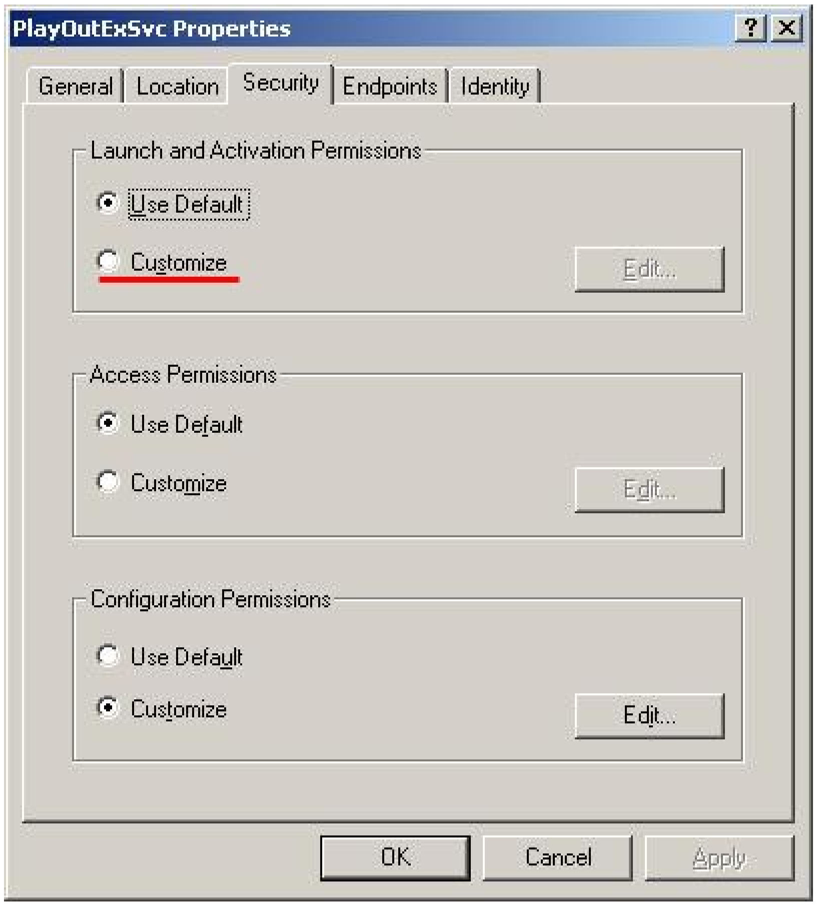
Permissions for Everyone	Allow	Deny
Local Launch	✓	
Remote Launch	✓	
Local Activation	✓	
Remote Activation	✓	

Then press Ok buttons to return to the "Component Services" window.

Then please select the "DCOM Config" folder and find there the PlayOutEventSvc component. Right click to this item and select "Properties" command from popup menu:



Please select the security tab in the Properties dialog and choose the Customize radiobutton in the “Launch and Activation Permission” section. The “Edit” button should become enabled.



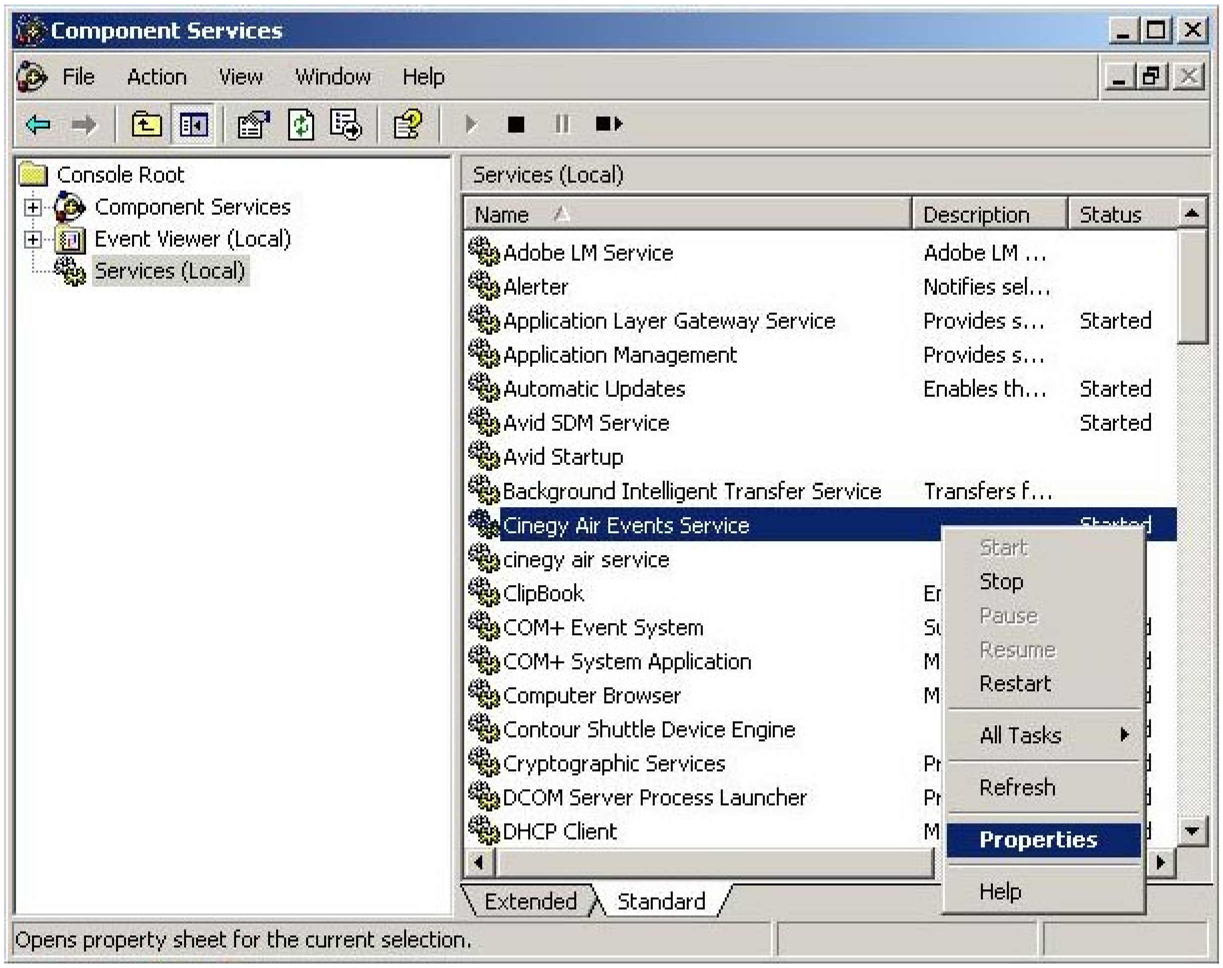
Launch and Activation Permissions
Use Default [✓]
Customize
Edit...

Access Permissions
Use Default [✓]
Customize
Edit...

Configuration Permissions
Use Default
Customize [✓]
Edit...

Press "Edit "button and set up the Launch Permissions exactly as it was made for "My Computer" folder.

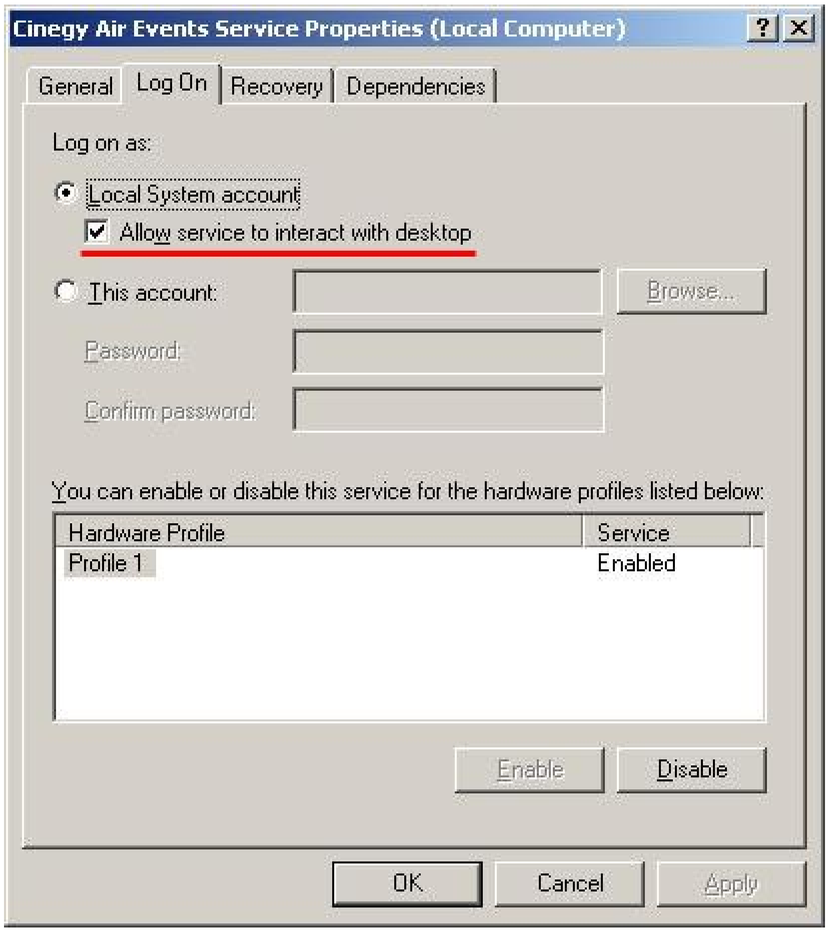
If you would like to see the console window during the event service is working, you have to configure the service itself. Please select the "Services" folder in the Components Services window and find the "Cinegy Air Event Service" in the right panel. Select the "Properties" command in the popup menu by the right click:



Name	Description	Status
Adobe LM Service	Adobe LM ...	
Alerter	Notifies sel...	
Application Layer Gateway Service	Provides s...	Started
Application Management	Provides s...	
Automatic Updates	Enables th...	Started
Avid SDM Service		Started
Avid Startup		
Background Intelligent Transfer Service	Transfers f...	
Cinegy Air Events Service		
cinegy air service		
ClipBook	Er	
COM+ Event System	Su	d
COM+ System Application	M	d
Computer Browser	M	d
Contour Shuttle Device Engine		d
Cryptographic Services	Pr	d
DCOM Server Process Launcher	Pr	d
DHCP Client	M	d

Opens property sheet for the current selection.

The Service Properties dialog appears. Please select the “Log On” tab and check the “Allow service to interact with desktop” option.


Cinegy Air Events Service Properties (Local Computer)

Log on as:

(●) Local System account

[✓] Allow service to interact with desktop

( ) This account:

Password:

Confirm password:

You can enable or disable this service for the hardware profiles listed below:

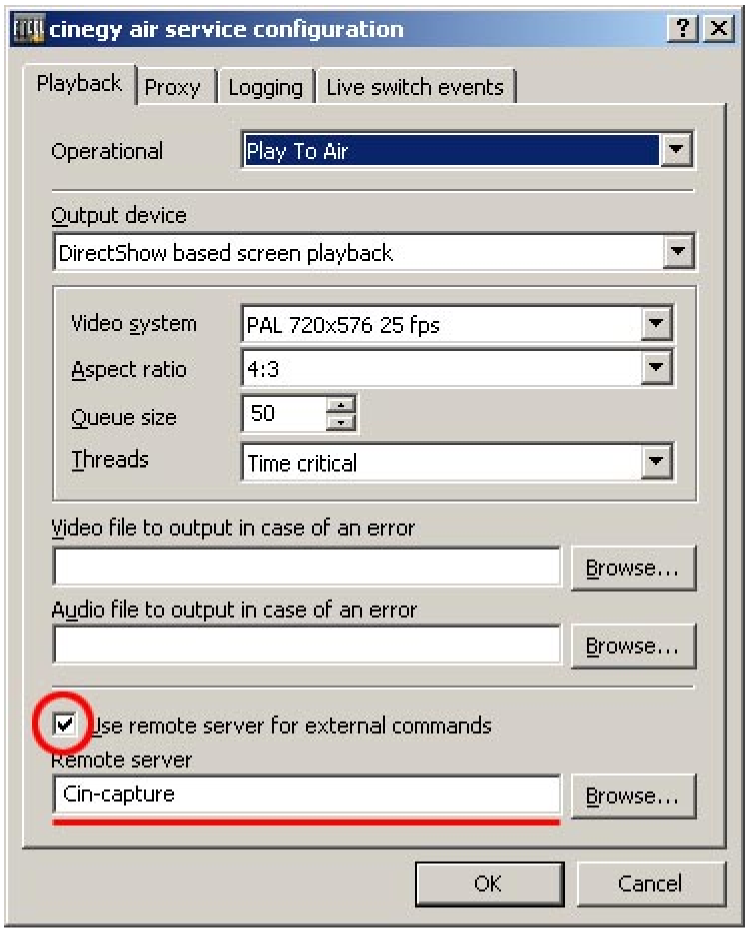
Hardware Profile	Service
Profile 1	Enabled

Finally please make sure that the Firewall protection is switched off.

After that the event service is configured and ready to be used.

On the Playout Servers

You have to configure each Playout Server, which should use the remote event service. You can use the Playout Configuration application for that. Just start “PlayOutExCfg.exe” utility from Playout folder and chose the corresponding remote PC (where event service is installed).



cinegy air service configuration

Playback | Proxy | Logging | Live switch events

Operational: Play To Air

Output device: DirectShow based screen playback

Video system: PAL 720x576 25 fps
Aspect ratio: 4:3
Queue size: 50
Threads: Time critical

Video file to output in case of an error: Browse...

Audio file to output in case of an error: Browse...

[✓] Use remote server for external commands

Remote server: Cin-capture Browse...

OK Cancel

Please make sure that “Use remote server” option is checked. You can use the “Browse…” command to find the corresponding server.

From now every event commands will be transferred to the remote event service to control the external device.

Event Handler Configuration Utility

Using this utility (EventsHandlerCfg.exe) you can select the devices you want to control and set up the connection parameters - like COM port and the “Minimal Commands Interval” (where supported). Using the last one you can limit the minimal time interval between sending two commands to the switcher. This option is quite important because the most of existing devices can not process two different commands, if they were sent just after each other. The right value you can find in the technical description of the device you use. Usually this should be set between 30 and 80 ms, though in some cases this could reach 3000 ms.

Controlling «VikinX VD0808» SDI switcher

Event commands format

Since you have installed the external event service as described above, you can use this service to control VikinX VD0808 video switcher. The table below shows you the events format is used to control this device.

Device	Command	Op1	Op2	Op3	Description
VIKINX	**crosspoint**	<Level>	<In>	<Out>	Set the crosspoint at the level <Level> from input <In> to output <Out>

For example, this event will connect the input 1 to the output 2 at the level 4:

Device: **VIKINX**

Command: **crosspoint**

Op1: 4

Op2: 1

Op3: 2

If this event is sent to the VikinX event service, the following message should appear in the service console window:

```
25.08.2006 12:00:32.107 EVENT: VIKINX crosspoint 4 1 2 : succeeded.
```

It means that service is successfully received the event.

Device configure

You can use the configuration utility to configure the device settings (e.g. COM port number you use to connect the «VikinX VD0808» device). This utility is automatically started by running InsatallLocalService.bat and InsatallRemoteService.bat files.

Controlling «Protocol 2000» based switcher (e.g. Kramer VS1616SDI)

Event commands format

The table below shows you the events format is used to control Protocol 2000 based devices.

Device	Command	Op1	Op2	Op3	Description
KRAMER	**crosspoint_video**	<Machine number>	<In>	<Out>	Set the video crosspoint only on the device defined by <Machine number> from video input <In> to video output <Out>
KRAMER	**crosspoint_audio**	<Machine number>	<In>	<Out>	Set the audio crosspoint only on the device defined by <Machine number> from audio input <In> to audio output <Out>
KRAMER	**crosspoint_all**	<Machine number>	<In>	<Out>	Set the both video and audio crosspoint on the device defined by <Machine number> from input <In> to output <Out>

GPI driven devices

GPI (General Purpose Interface) is the universal interface which can be used to control any external devices which support the GPI protocol. Cinegy supports the SeaLevel GPI board, which should be installed in order to use the GPI signaling.

Event commands format

The table below shows you the events format is used to control the GPI driven devices.

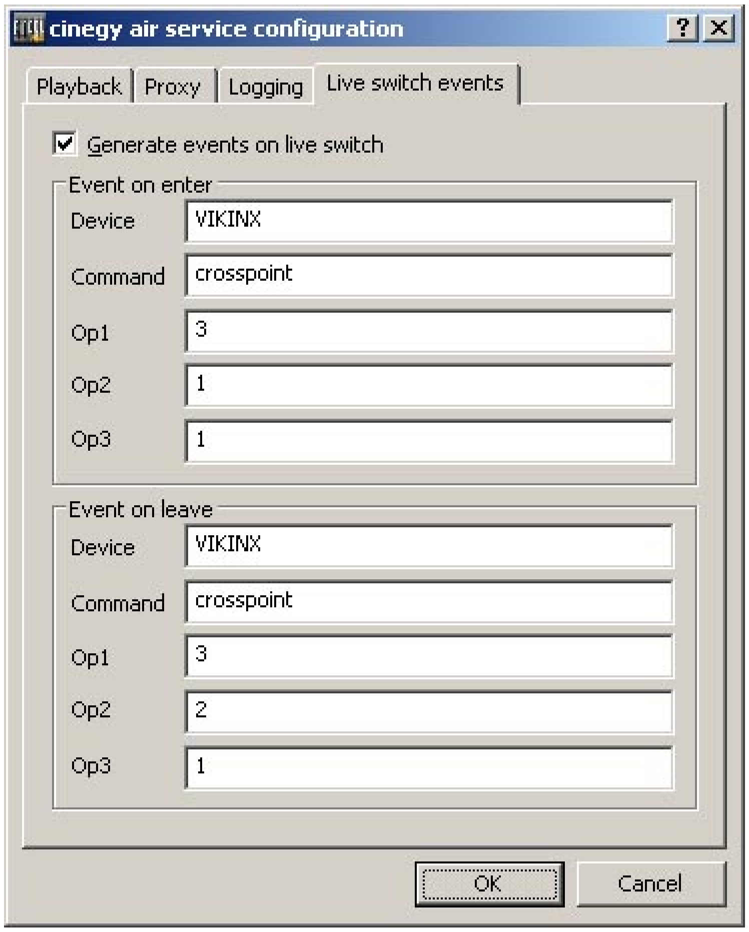
Device	Command	Op1	Op2	Op3	Description
GPI	**setpin**	<pin number> [0..7]	<value> [0 or 1]	-	Set the value (0 or 1) on the specified GPI pin.
GPI	**setbyte**	<byte mask> [0..255]	-	-	Set all the pins by mask.

Using automatic live mode switching

If you need to use the external device to switch between playout and external live signal, you can make use of the special possibility: you can define events, which will be sent automatically at the beginning and ending of each live item.

You can do it in the Playout Configuration utility. Please start “PlayOutExCfg.exe” on the corresponding playout server and select “Live switch events” tab. Then just fill out the corresponding event commands for enter and leave the live mode. Please make sure that “Generate event on live switch” is checked.

The configuration in the example below automatically connects the input 1 to the output 1 at the level 3 for the live items and input 2 to the output 1 for the items, which should be played by playout server (this example is for the VikinX switcher case).



[✓] Generate events on live switch

Event on enter

Device	VIKINX
Command	crosspoint
Op1	3
Op2	1
Op3	1

Event on leave

Device	VIKINX
Command	crosspoint
Op1	3
Op2	2
Op3	1

Samples

In the PlayOutEventsSvc folder there is a sample of the event handler implementation (C++, ATL). It is registered as a service and implements the IPlayOutExEventSink interface. This sample outputs the event text and parameters to the console window.

In the EventServiceSample folder there are the sample implementation of the handler plug-in with the Event Service and the plug-in setup utility. Launch Register.bat from the "_install" directory. The plug-in project is in «SampleEventPlugin» directory. After building the project launch the setup utility EventsHandlerCfg.exe (from the «_install» directory) - the Sample Event Plugin will appear. Select it and tick off "Activate". After this all events will be sent to this plug-in.

EventSvcManager.exe is a main Events Service which receives events from the Air Service and transmits them to the event plug-ins.

In this way it is possible to add another handler plug-ins.

The project is builded in Visual Studio 2005.