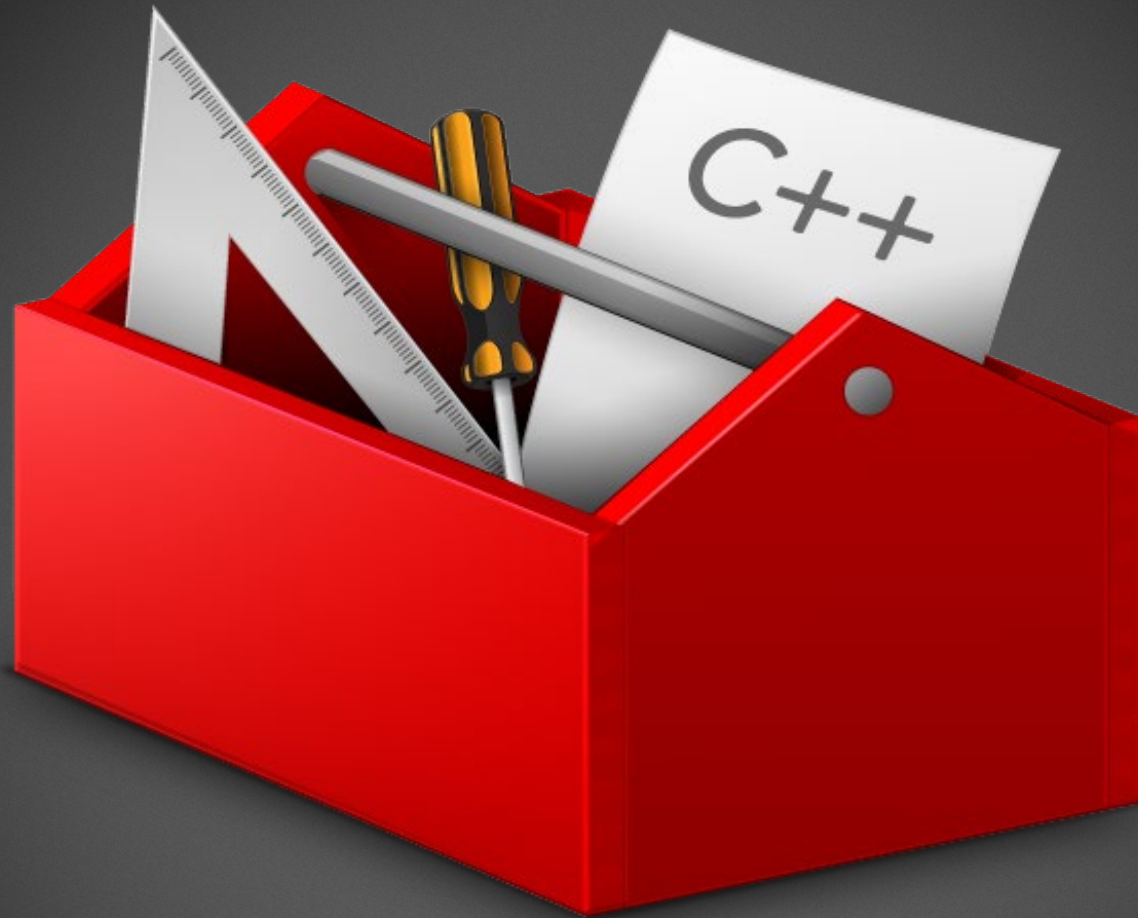


# SDK-ATEM Switchers Software Developers Kit.

Blackmagicdesign 



Mac OS X™

Windows™

December 2015

# Table of Contents

---

<b>1</b>	<b>Introduction</b>	<b>27</b>
1.1	Welcome	27
1.2	Overview	27
1.3	API Design	28
1.3.1	Overview	28
1.3.2	Object Model	28
1.3.3	Object Interfaces	28
1.3.4	Reference Counting	29
1.3.5	Interface Stability	29
1.3.5.1	New Interfaces	29
1.3.5.2	Updated Interfaces	30
1.3.5.3	Deprecated Interfaces	30
1.3.5.4	Removed interfaces	30
1.4	Interface Reference	31
1.4.1	IUnknown Interface	31
1.4.1.1	IUnknown::QueryInterface method	32
1.4.1.2	IUnknown::AddRef method	33
1.4.1.3	IUnknown::Release method	33
1.5	Using the Switcher API in a project	34
1.5.1	Basic Types	35
1.5.2	Accessing Switcher devices	36
1.5.2.1	Windows	36
1.5.2.2	Mac OS X	37
<b>2</b>	<b>Basic Switcher Control</b>	<b>38</b>
2.1	General Information	38
2.1.1	Switcher Configuration and Transitions	38
2.1.2	Switcher Interface Diagram	39
2.2	Switcher Data Types	40
2.2.1	Basic Switcher Data Types	40
2.2.2	Switcher Event Type	40
2.2.3	Switcher Power Status	40
2.2.4	Switcher Video Mode	41

# Table of Contents

---

2.2.5	Switcher Down Conversion Methods	42
2.2.6	Switcher Input Event Types	42
2.2.7	Switcher External Port Types	43
2.2.8	Switcher Port Types	43
2.2.9	Switcher Input Availability	44
2.2.10	Switcher Mix Effect Block Properties	44
2.2.11	Switcher Connection Errors	45
2.2.12	Switcher MultiView Layouts	45
2.2.13	Switcher Serial Port Functions	45
2.2.14	Switcher Color Events	46
2.2.15	Switcher Aux Events	46
2.2.16	Switcher MultiView Events	46
2.2.17	Switcher Serial Port Event Types	46
2.3	Interface Reference	47
2.3.1	IBMDSwitcherDiscovery Interface	47
2.3.1.1	IBMDSwitcherDiscovery::ConnectTo method	48
2.3.2	IBMDSwitcher Interface	49
2.3.2.1	IBMDSwitcher:: GetProductName method	50
2.3.2.2	IBMDSwitcher:: GetVideoMode method	50
2.3.2.3	IBMDSwitcher:: SetVideoMode method	51
2.3.2.4	IBMDSwitcher:: DoesSupportVideoMode method	52
2.3.2.5	IBMDSwitcher:: GetMethodForDownConvertedSD method	53
2.3.2.6	IBMDSwitcher:: SetMethodForDownConvertedSD method	54
2.3.2.7	IBMDSwitcher:: GetDownConvertedHDVideoMode method	55
2.3.2.8	IBMDSwitcher:: SetDownConvertedHDVideoMode method	56
2.3.2.9	IBMDSwitcher:: DoesSupportDownConvertedHDVideoMode method	57
2.3.2.10	IBMDSwitcher:: GetMultiViewVideoMode method	58
2.3.2.11	IBMDSwitcher:: SetMultiViewVideoMode method	59
2.3.2.12	IBMDSwitcher:: DoesSupportMultiViewVideoMode method	60
2.3.2.13	IBMDSwitcher:: GetPowerStatus method	61
2.3.2.14	IBMDSwitcher:: CreateIterator method	62
2.3.2.15	IBMDSwitcher::AddCallback method	63
2.3.2.16	IBMDSwitcher::RemoveCallback method	64
2.3.3	IBMDSwitcherCallback Interface	65
2.3.3.1	IBMDSwitcherCallback::Notify method	66
2.3.4	IBMDSwitcherInputIterator Interface	67

# Table of Contents

---

2.3.4.1	IBMDSwitcherInputIterator::Next method	68
2.3.4.2	IBMDSwitcherInputIterator::GetById method	69
2.3.5	IBMDSwitcherInput Interface	70
2.3.5.1	IBMDSwitcherInput::AddCallback method	72
2.3.5.2	IBMDSwitcherInput::RemoveCallback method	73
2.3.5.3	IBMDSwitcherInput::GetInputId method	73
2.3.5.4	IBMDSwitcherInput::GetPortType method	74
2.3.5.5	IBMDSwitcherInput::GetInputAvailability method	75
2.3.5.6	IBMDSwitcherInput::SetShortName method	76
2.3.5.7	IBMDSwitcherInput::GetShortName method	77
2.3.5.8	IBMDSwitcherInput::SetLongName method	78
2.3.5.9	IBMDSwitcherInput::GetLongName method	79
2.3.5.10	IBMDSwitcherInput::ResetNames method	80
2.3.5.11	IBMDSwitcherInput::IsProgramTallied method	81
2.3.5.12	IBMDSwitcherInput::IsPreviewTallied method	82
2.3.5.13	IBMDSwitcherInput::GetAvailableExternalPortTypes method	83
2.3.5.14	IBMDSwitcherInput::SetCurrentExternalPortType method	84
2.3.6	IBMDSwitcherInputCallback Interface	85
2.3.6.1	IBMDSwitcherInputCallback::Notify method	86
2.3.7	IBMDSwitcherMixEffectBlockIterator Interface	87
2.3.7.1	IBMDSwitcherMixEffectBlockIterator::Next method	88
2.3.8	IBMDSwitcherMixEffectBlock Interface	89
2.3.8.1	IBMDSwitcherMixEffectBlock::CreateIterator method	90
2.3.8.2	IBMDSwitcherMixEffectBlock::AddCallback method	91
2.3.8.3	IBMDSwitcherMixEffectBlock::RemoveCallback method	92
2.3.8.4	IBMDSwitcherMixEffectBlock::GetFlag method	93
2.3.8.5	IBMDSwitcherMixEffectBlock::GetInt method	94
2.3.8.6	IBMDSwitcherMixEffectBlock::GetFloat method	95
2.3.8.7	IBMDSwitcherMixEffectBlock::GetString method	96
2.3.8.8	IBMDSwitcherMixEffectBlock::SetFlag method	97
2.3.8.9	IBMDSwitcherMixEffectBlock::SetInt method	98
2.3.8.10	IBMDSwitcherMixEffectBlock::SetFloat method	99
2.3.8.11	IBMDSwitcherMixEffectBlock::SetString method	100
2.3.8.12	IBMDSwitcherMixEffectBlock::PerformAutoTransition method	101
2.3.8.13	IBMDSwitcherMixEffectBlock::PerformCut method	102
2.3.8.14	IBMDSwitcherMixEffectBlock::PerformFadeToBlack method	102

# Table of Contents

---

2.3.9	IBMDSwitcherMixEffectBlockCallback Interface	103
2.3.9.1	IBMDSwitcherMixEffectBlockCallback::PropertyChanged method	104
2.3.10	IBMDSwitcherInputColor Interface	105
2.3.10.1	IBMDSwitcherInputColor::GetHue method	106
2.3.10.2	IBMDSwitcherInputColor::SetHue method	107
2.3.10.3	IBMDSwitcherInputColor::GetSaturation method	107
2.3.10.4	IBMDSwitcherInputColor::SetSaturation method	108
2.3.10.5	IBMDSwitcherInputColor::GetLuma method	108
2.3.10.6	IBMDSwitcherInputColor::SetLuma method	109
2.3.10.7	IBMDSwitcherInputColor::AddCallback method	110
2.3.10.8	IBMDSwitcherInputColor::RemoveCallback method	111
2.3.11	IBMDSwitcherInputColorCallback Interface	112
2.3.11.1	IBMDSwitcherInputColorCallback::Notify method	113
2.3.12	IBMDSwitcherInputAux Interface	114
2.3.12.1	IBMDSwitcherInputAux::GetInputSource method	115
2.3.12.2	IBMDSwitcherInputAux::SetInputSource method	116
2.3.12.3	IBMDSwitcherInputAux::GetInputAvailabilityMask method	117
2.3.12.4	IBMDSwitcherInputAux::AddCallback method	118
2.3.12.5	IBMDSwitcherInputAux::RemoveCallback method	119
2.3.13	IBMDSwitcherInputAuxCallback Interface	120
2.3.13.1	IBMDSwitcherInputAuxCallback::Notify method	121
2.3.14	IBMDSwitcherMultiViewIterator Interface	122
2.3.14.1	IBMDSwitcherMultiViewIterator::Next method	123
2.3.15	IBMDSwitcherMultiView Interface	124
2.3.15.1	IBMDSwitcherMultiView::GetLayout method	125
2.3.15.2	IBMDSwitcherMultiView::SetLayout method	126
2.3.15.3	IBMDSwitcherMultiView::GetWindowInput method	127
2.3.15.4	IBMDSwitcherMultiView::SetWindowInput method	128
2.3.15.5	IBMDSwitcherMultiView::GetWindowCount method	129
2.3.15.6	IBMDSwitcherMultiView::GetInputAvailabilityMask method	130
2.3.15.7	IBMDSwitcherMultiView::CanRouteInputs method	131
2.3.15.8	IBMDSwitcherMultiView::AddCallback method	132
2.3.15.9	IBMDSwitcherMultiView::RemoveCallback method	133
2.3.16	IBMDSwitcherMultiViewCallback Interface	134
2.3.16.1	IBMDSwitcherMultiViewCallback::Notify method	135
2.3.17	IBMDSwitcherSerialPortIterator Interface	136

# Table of Contents

2.3.17.1	IBMDSwitcherSerialPortIterator::Next method	137
2.3.18	IBMDSwitcherSerialPort Interface	138
2.3.18.1	IBMDSwitcherSerialPort::SetFunction method	139
2.3.18.2	IBMDSwitcherSerialPort::GetFunction method	140
2.3.18.3	IBMDSwitcherSerialPort::DoesSupportFunction method	141
2.3.18.4	IBMDSwitcherSerialPort::AddCallback method	142
2.3.18.5	IBMDSwitcherSerialPort::RemoveCallback method	143
2.3.19	IBMDSwitcherSerialPortCallback Interface	144
2.3.19.1	IBMDSwitcherSerialPortCallback::Notify method	145

## 3 Advanced Transitions 146

3.1	Data Types	146
3.1.1	Mix Parameters Event Type	146
3.1.2	Dip Parameters Event Type	146
3.1.3	Wipe Parameters Event Type	146
3.1.4	DVE Parameters Event Type	147
3.1.5	Stinger Parameters Event Type	148
3.1.6	Transition Parameters Event Type	148
3.1.7	Transition Style	149
3.1.8	Transition Selection	149
3.1.9	DVE Transition Style	150
3.1.10	Stinger Transition Source	151
3.2	Interface Reference	152
3.2.1	IBMDSwitcherTransitionMixParameters Interface	152
3.2.1.1	IBMDSwitcherTransitionMixParameters::GetRate method	153
3.2.1.2	IBMDSwitcherTransitionMixParameters::SetRate method	154
3.2.1.3	IBMDSwitcherTransitionMixParameters::AddCallback method	155
3.2.1.4	IBMDSwitcherTransitionMixParameters::RemoveCallback method	156
3.2.2	IBMDSwitcherTransitionMixParametersCallback Interface	157
3.2.2.1	IBMDSwitcherTransitionMixParametersCallback::Notify method	158
3.2.3	IBMDSwitcherTransitionDipParameters Interface	159
3.2.3.1	IBMDSwitcherTransitionDipParameters::GetRate method	160
3.2.3.2	IBMDSwitcherTransitionDipParameters::SetRate method	161
3.2.3.3	IBMDSwitcherTransitionDipParameters::GetInputDip method	162
3.2.3.4	IBMDSwitcherTransitionDipParameters::SetInputDip method	163



# Table of Contents

---

3.2.3.5	IBMDSwitcherTransitionDipParameters::AddCallback method	164
3.2.3.6	IBMDSwitcherTransitionDipParameters::RemoveCallback method	165
3.2.4	IBMDSwitcherTransitionDipParametersCallback Interface	166
3.2.4.1	IBMDSwitcherTransitionDipParametersCallback::Notify method	167
3.2.5	IBMDSwitcherTransitionWipeParametersCallback Interface	168
3.2.5.1	IBMDSwitcherTransitionWipeParametersCallback::Notify method	169
3.2.6	IBMDSwitcherTransitionWipeParameters Interface	170
3.2.6.1	IBMDSwitcherTransitionWipeParameters::GetRate method	172
3.2.6.2	IBMDSwitcherTransitionWipeParameters::SetRate method	173
3.2.6.3	IBMDSwitcherTransitionWipeParameters::GetPattern method	174
3.2.6.4	IBMDSwitcherTransitionWipeParameters::SetPattern method	175
3.2.6.5	IBMDSwitcherTransitionWipeParameters::GetBorderSize method	176
3.2.6.6	IBMDSwitcherTransitionWipeParameters::SetBorderSize method	176
3.2.6.7	IBMDSwitcherTransitionWipeParameters::GetInputBorder method	177
3.2.6.8	IBMDSwitcherTransitionWipeParameters::SetInputBorder method	178
3.2.6.9	IBMDSwitcherTransitionWipeParameters::GetSymmetry method	179
3.2.6.10	IBMDSwitcherTransitionWipeParameters::SetSymmetry method	179
3.2.6.11	IBMDSwitcherTransitionWipeParameters::GetSoftness method	180
3.2.6.12	IBMDSwitcherTransitionWipeParameters::SetSoftness method	180
3.2.6.13	IBMDSwitcherTransitionWipeParameters::GetHorizontalOffset method	181
3.2.6.14	IBMDSwitcherTransitionWipeParameters::SetHorizontalOffset method	181
3.2.6.15	IBMDSwitcherTransitionWipeParameters::GetVerticalOffset method	182
3.2.6.16	IBMDSwitcherTransitionWipeParameters::SetVerticalOffset method	182
3.2.6.17	IBMDSwitcherTransitionWipeParameters::GetReverse method	183
3.2.6.18	IBMDSwitcherTransitionWipeParameters::SetReverse method	183
3.2.6.19	IBMDSwitcherTransitionWipeParameters::GetFlipFlop method	184
3.2.6.20	IBMDSwitcherTransitionWipeParameters::SetFlipFlop method	184
3.2.6.21	IBMDSwitcherTransitionWipeParameters::AddCallback method	185
3.2.6.22	IBMDSwitcherTransitionWipeParameters::RemoveCallback method	186
3.2.7	IBMDSwitcherTransitionDVEParameters Interface	187
3.2.7.1	IBMDSwitcherTransitionDVEParameters::GetRate method	189
3.2.7.2	IBMDSwitcherTransitionDVEParameters::SetRate method	190
3.2.7.3	IBMDSwitcherTransitionDVEParameters::GetLogoRate method	191
3.2.7.4	IBMDSwitcherTransitionDVEParameters::SetLogoRate method	192
3.2.7.5	IBMDSwitcherTransitionDVEParameters::GetReverse method	193
3.2.7.6	IBMDSwitcherTransitionDVEParameters::SetReverse method	193

# Table of Contents

---

3.2.7.7	IBMDSwitcherTransitionDVEParameters::GetFlipFlop method	194
3.2.7.8	IBMDSwitcherTransitionDVEParameters::SetFlipFlop method	194
3.2.7.9	IBMDSwitcherTransitionDVEParameters::GetStyle method	195
3.2.7.10	IBMDSwitcherTransitionDVEParameters::SetStyle method	196
3.2.7.11	IBMDSwitcherTransitionDVEParameters::GetInputFill method	197
3.2.7.12	IBMDSwitcherTransitionDVEParameters::SetInputFill method	198
3.2.7.13	IBMDSwitcherTransitionDVEParameters::GetInputCut method	199
3.2.7.14	IBMDSwitcherTransitionDVEParameters::SetInputCut method	200
3.2.7.15	IBMDSwitcherTransitionDVEParameters::GetFillInputAvailabilityMask method	201
3.2.7.16	IBMDSwitcherTransitionDVEParameters::GetCutInputAvailabilityMask method	202
3.2.7.17	IBMDSwitcherTransitionDVEParameters::GetEnableKey method	203
3.2.7.18	IBMDSwitcherTransitionDVEParameters::SetEnableKey method	203
3.2.7.19	IBMDSwitcherTransitionDVEParameters::GetPreMultiplied method	204
3.2.7.20	IBMDSwitcherTransitionDVEParameters::SetPreMultiplied method	204
3.2.7.21	IBMDSwitcherTransitionDVEParameters::GetClip method	205
3.2.7.22	IBMDSwitcherTransitionDVEParameters::SetClip method	205
3.2.7.23	IBMDSwitcherTransitionDVEParameters::GetGain method	206
3.2.7.24	IBMDSwitcherTransitionDVEParameters::SetGain method	206
3.2.7.25	IBMDSwitcherTransitionDVEParameters::GetInverse method	207
3.2.7.26	IBMDSwitcherTransitionDVEParameters::SetInverse method	207
3.2.7.27	IBMDSwitcherTransitionDVEParameters::AddCallback method	208
3.2.7.28	IBMDSwitcherTransitionDVEParameters::RemoveCallback method	209
3.2.8	IBMDSwitcherTransitionDVEParametersCallback Interface	210
3.2.8.1	IBMDSwitcherTransitionDVEParametersCallback::Notify method	211
3.2.9	IBMDSwitcherTransitionStingerParameters Interface	212
3.2.9.1	IBMDSwitcherTransitionStingerParameters::GetSource method	214
3.2.9.2	IBMDSwitcherTransitionStingerParameters::SetSource method	215
3.2.9.3	IBMDSwitcherTransitionStingerParameters::GetPreMultiplied method	216
3.2.9.4	IBMDSwitcherTransitionStingerParameters::SetPreMultiplied method	216
3.2.9.5	IBMDSwitcherTransitionStingerParameters::GetClip method	217
3.2.9.6	IBMDSwitcherTransitionStingerParameters::SetClip method	217
3.2.9.7	IBMDSwitcherTransitionStingerParameters::GetGain method	218
3.2.9.8	IBMDSwitcherTransitionStingerParameters::SetGain method	218
3.2.9.9	IBMDSwitcherTransitionStingerParameters::GetInverse method	219
3.2.9.10	IBMDSwitcherTransitionStingerParameters::SetInverse method	219
3.2.9.11	IBMDSwitcherTransitionStingerParameters::GetPreroll method	220



# Table of Contents

3.2.9.12	IBMDSwitcherTransitionStingerParameters::SetPreroll method	220
3.2.9.13	IBMDSwitcherTransitionStingerParameters::GetClipDuration method	221
3.2.9.14	IBMDSwitcherTransitionStingerParameters::SetClipDuration method	221
3.2.9.15	IBMDSwitcherTransitionStingerParameters::GetTriggerPoint method	222
3.2.9.16	IBMDSwitcherTransitionStingerParameters::SetTriggerPoint method	222
3.2.9.17	IBMDSwitcherTransitionStingerParameters::GetMixRate method	223
3.2.9.18	IBMDSwitcherTransitionStingerParameters::SetMixRate method	223
3.2.9.19	IBMDSwitcherTransitionStingerParameters::AddCallback method	224
3.2.9.20	IBMDSwitcherTransitionStingerParameters::RemoveCallback method	225
3.2.10	IBMDSwitcherTransitionStingerParametersCallback Interface	226
3.2.10.1	IBMDSwitcherTransitionStingerParametersCallback::Notify method	227
3.2.11	IBMDSwitcherTransitionParameters Interface	228
3.2.11.1	IBMDSwitcherTransitionParameters::GetTransitionStyle method	229
3.2.11.2	IBMDSwitcherTransitionParameters::GetNextTransitionStyle method	229
3.2.11.3	IBMDSwitcherTransitionParameters::SetNextTransitionStyle method	230
3.2.11.4	IBMDSwitcherTransitionParameters::GetTransitionSelection method	231
3.2.11.5	IBMDSwitcherTransitionParameters::SetNextTransitionSelection method	232
3.2.11.6	IBMDSwitcherTransitionParameters::GetNextTransitionSelection method	233
3.2.11.7	IBMDSwitcherTransitionParameters::AddCallback method	234
3.2.11.8	IBMDSwitcherTransitionParameters::RemoveCallback method	235
3.2.12	IBMDSwitcherTransitionParametersCallback Interface	236
3.2.12.1	IBMDSwitcherTransitionParametersCallback::Notify method	237

## 4 Switcher Media 238

4.1	General Information	238
4.1.1	Uploading a Still or Clip	238
4.1.2	Downloading a Still or Clip	239
4.2	Media Data Types	240
4.2.1	Switcher Pixel Format	240
4.2.2	Media Player Source Type	240
4.2.3	Media Pool Event Type	240
4.3	Interface Reference	241
4.3.1	IBMDSwitcherMediaPlayerCallback Interface	241
4.3.1.1	IBMDSwitcherMediaPlayerCallback::SourceChanged method	242
4.3.1.2	IBMDSwitcherMediaPlayerCallback::PlayingChanged method	242

# Table of Contents

---

4.3.1.3	IBMDSwitcherMediaPlayerCallback::LoopChanged method	243
4.3.1.4	IBMDSwitcherMediaPlayerCallback::AtBeginningChanged method	243
4.3.1.5	IBMDSwitcherMediaPlayerCallback::ClipFrameChanged method	244
4.3.2	IBMDSwitcherMediaPlayerIterator Interface	245
4.3.2.1	IBMDSwitcherMediaPlayerIterator::Next method	246
4.3.3	IBMDSwitcherMediaPlayer Interface	247
4.3.3.1	IBMDSwitcherMediaPlayer::GetSource method	248
4.3.3.2	IBMDSwitcherMediaPlayer::SetSource method	249
4.3.3.3	IBMDSwitcherMediaPlayer::GetPlaying method	250
4.3.3.4	IBMDSwitcherMediaPlayer::SetPlaying method	250
4.3.3.5	IBMDSwitcherMediaPlayer::GetLoop method	251
4.3.3.6	IBMDSwitcherMediaPlayer::SetLoop method	251
4.3.3.7	IBMDSwitcherMediaPlayer::GetAtBeginning method	252
4.3.3.8	IBMDSwitcherMediaPlayer::SetAtBeginning method	253
4.3.3.9	IBMDSwitcherMediaPlayer::GetClipFrame method	253
4.3.3.10	IBMDSwitcherMediaPlayer::SetClipFrame method	254
4.3.3.11	IBMDSwitcherMediaPlayer::AddCallback method	255
4.3.3.12	IBMDSwitcherMediaPlayer::RemoveCallback method	256
4.3.4	IBMDSwitcherFrame Interface	257
4.3.4.1	IBMDSwitcherFrame::GetWidth method	258
4.3.4.2	IBMDSwitcherFrame::GetHeight method	258
4.3.4.3	IBMDSwitcherFrame::GetRowBytes method	259
4.3.4.4	IBMDSwitcherFrame::GetPixelFormat method	259
4.3.4.5	IBMDSwitcherFrame::GetBytes method	260
4.3.5	IBMDSwitcherAudio Interface	261
4.3.5.1	IBMDSwitcherFrame::GetSize method	262
4.3.5.2	IBMDSwitcherAudio::GetBytes method	262
4.3.6	IBMDSwitcherLockCallback Interface	263
4.3.6.1	IBMDSwitcherLockCallback::Obtained method	264
4.3.7	IBMDSwitcherStillsCallback Interface	265
4.3.7.1	IBMDSwitcherStillsCallback::Notify method	266
4.3.8	IBMDSwitcherStills Interface	267
4.3.8.1	IBMDSwitcherStills::GetCount method	268
4.3.8.2	IBMDSwitcherStills::IsValid method	268
4.3.8.3	IBMDSwitcherStills::GetName method	269
4.3.8.4	IBMDSwitcherStills::SetName method	270

# Table of Contents

---

4.3.8.5	IBMDSwitcherStills::GetHash method	271
4.3.8.6	IBMDSwitcherStills::SetInvalid method	272
4.3.8.7	IBMDSwitcherStills::Lock method	273
4.3.8.8	IBMDSwitcherStills::Unlock method	274
4.3.8.9	IBMDSwitcherStills::Upload method	275
4.3.8.10	IBMDSwitcherStills::Download method	276
4.3.8.11	IBMDSwitcherStills::CancelTransfer method	277
4.3.8.12	IBMDSwitcherStills::GetProgress method	277
4.3.8.13	IBMDSwitcherStills::AddCallback method	278
4.3.8.14	IBMDSwitcherStills::RemoveCallback method	279
4.3.9	IBMDSwitcherClipCallback Interface	280
4.3.9.1	IBMDSwitcherClipCallback::Notify method	281
4.3.10	IBMDSwitcherClip Interface	283
4.3.10.1	IBMDSwitcherClip::GetIndex method	285
4.3.10.2	IBMDSwitcherClip::IsValid method	285
4.3.10.3	IBMDSwitcherClip::GetName method	286
4.3.10.4	IBMDSwitcherClip::SetName method	287
4.3.10.5	IBMDSwitcherClip::SetValid method	288
4.3.10.6	IBMDSwitcherClip::SetInvalid method	289
4.3.10.7	IBMDSwitcherClip::GetFrameCount method	290
4.3.10.8	IBMDSwitcherClip::GetMaxFrameCount method	291
4.3.10.9	IBMDSwitcherClip::IsFrameValid method	292
4.3.10.10	IBMDSwitcherClip::GetFrameHash method	293
4.3.10.11	IBMDSwitcherClip::IsAudioValid method	294
4.3.10.12	IBMDSwitcherClip::GetAudioName method	295
4.3.10.13	IBMDSwitcherClip::SetAudioName method	296
4.3.10.14	IBMDSwitcherClip::GetAudioHash method	297
4.3.10.15	IBMDSwitcherClip::SetAudioInvalid method	297
4.3.10.16	IBMDSwitcherClip::Lock method	298
4.3.10.17	IBMDSwitcherClip::Unlock method	299
4.3.10.18	IBMDSwitcherClip::UploadFrame method	300
4.3.10.19	IBMDSwitcherClip::DownloadFrame method	301
4.3.10.20	IBMDSwitcherClip::UploadAudio method	302
4.3.10.21	IBMDSwitcherClip::DownloadAudio method	303
4.3.10.22	IBMDSwitcherClip::CancelTransfer method	304
4.3.10.23	IBMDSwitcherClip::GetProgress method	304

# Table of Contents

4.3.10.24	IBMDSwitcherClip::AddCallback method	305
4.3.10.25	IBMDSwitcherClip::RemoveCallback method	306
4.3.11	IBMDSwitcherMediaPoolCallback Interface	307
4.3.11.1	IBMDSwitcherMediaPoolCallback::ClipFrameMaxCountsChanged method	308
4.3.11.2	IBMDSwitcherMediaPoolCallback::FrameTotalForClipsChanged method	308
4.3.12	IBMDSwitcherMediaPool Interface	309
4.3.12.1	IBMDSwitcherMediaPool::GetStills method	310
4.3.12.2	IBMDSwitcherMediaPool::GetClip method	311
4.3.12.3	IBMDSwitcherMediaPool::GetClipCount method	312
4.3.12.4	IBMDSwitcherMediaPool::CreateFrame method	313
4.3.12.5	IBMDSwitcherMediaPool::CreateAudio method	314
4.3.12.6	IBMDSwitcherMediaPool::GetFrameTotalForClips method	315
4.3.12.7	IBMDSwitcherMediaPool::GetClipMaxFrameCounts method	316
4.3.12.8	IBMDSwitcherMediaPool::Clear method	317
4.3.12.9	IBMDSwitcherMediaPool::SetClipMaxFrameCounts method	318
4.3.12.10	IBMDSwitcherMediaPool::AddCallback method	319
4.3.12.11	IBMDSwitcherMediaPool::RemoveCallback method	320

## 5    **Keys**    **321**

5.1	Key Data Types	321
5.1.1	Key Type	321
5.1.2	Fly Key Frames	321
5.1.3	Border Bevel Options	322
5.1.4	Key Event Type	322
5.1.5	Luminance Key Parameters Event Type	323
5.1.6	Chroma Key Parameters Event Type	323
5.1.7	Pattern Key Parameters Event Type	323
5.1.8	DVE Key Parameters Event Type	324
5.1.9	Fly Key Parameters Event Type	325
5.1.10	Downstream Key Event Type	326
5.2	Interface Reference	327
5.2.1	IBMDSwitcherKeyIterator Interface	327
5.2.1.1	IBMDSwitcherMediaPool::AddCallback method	328
5.2.2	IBMDSwitcherKey Interface	329
5.2.2.1	IBMDSwitcherKey::GetType method	331

# Table of Contents

---

5.2.2.2	IBMDSwitcherKey::SetType method	332
5.2.2.3	IBMDSwitcherKey::GetInputCut method	333
5.2.2.4	IBMDSwitcherKey::SetInputCut method	334
5.2.2.5	IBMDSwitcherKey::GetInputFill method	335
5.2.2.6	IBMDSwitcherKey::SetInputFill method	336
5.2.2.7	IBMDSwitcherKey::GetFillInputAvailabilityMask method	337
5.2.2.8	IBMDSwitcherKey::GetCutInputAvailabilityMask method	338
5.2.2.9	IBMDSwitcherKey::GetOnAir method	339
5.2.2.10	IBMDSwitcherKey::SetOnAir method	339
5.2.2.11	IBMDSwitcherKey::CanBeDVEKey method	340
5.2.2.12	IBMDSwitcherKey::GetMasked method	341
5.2.2.13	IBMDSwitcherKey::SetMasked method	341
5.2.2.14	IBMDSwitcherKey::GetMaskTop method	342
5.2.2.15	IBMDSwitcherKey::SetMaskTop method	342
5.2.2.16	IBMDSwitcherKey::GetMaskBottom method	343
5.2.2.17	IBMDSwitcherKey::SetMaskBottom method	343
5.2.2.18	IBMDSwitcherKey::GetMaskLeft method	344
5.2.2.19	IBMDSwitcherKey::SetMaskLeft method	344
5.2.2.20	IBMDSwitcherKey::GetMaskRight method	345
5.2.2.21	IBMDSwitcherKey::SetMaskRight method	345
5.2.2.22	IBMDSwitcherKey::ResetMask method	346
5.2.2.23	IBMDSwitcherKey::GetTransitionSelectionMask method	346
5.2.2.24	IBMDSwitcherKey::AddCallback method	347
5.2.2.25	IBMDSwitcherKey::RemoveCallback method	348
5.2.3	IBMDSwitcherKeyCallback Interface	349
5.2.3.1	IBMDSwitcherKeyCallback::Notify method	350
5.2.4	IBMDSwitcherKeyLumaParameters Interface	351
5.2.4.1	IBMDSwitcherKeyLumaParameters::GetPreMultiplied method	352
5.2.4.2	IBMDSwitcherKeyLumaParameters::SetPreMultiplied method	353
5.2.4.3	IBMDSwitcherKeyLumaParameters::GetClip method	354
5.2.4.4	IBMDSwitcherKeyLumaParameters::SetClip method	354
5.2.4.5	IBMDSwitcherKeyLumaParameters::GetGain method	355
5.2.4.6	IBMDSwitcherKeyLumaParameters::SetGain method	355
5.2.4.7	IBMDSwitcherKeyLumaParameters::GetInverse method	356
5.2.4.8	IBMDSwitcherKeyLumaParameters::SetInverse method	356
5.2.4.9	IBMDSwitcherKeyLumaParameters::AddCallback method	357

# Table of Contents

---

5.2.4.10	IBMDSwitcherKeyLumaParameters::RemoveCallback method	358
5.2.5	IBMDSwitcherKeyLumaParametersCallback Interface	359
5.2.5.1	IBMDSwitcherKeyLumaParametersCallback::Notify method	360
5.2.6	IBMDSwitcherKeyChromaParameters Interface	361
5.2.6.1	IBMDSwitcherKeyChromaParameters::GetHue method	362
5.2.6.2	IBMDSwitcherKeyChromaParameters::SetHue method	362
5.2.6.3	IBMDSwitcherKeyChromaParameters::GetGain method	363
5.2.6.4	IBMDSwitcherKeyChromaParameters::SetGain method	363
5.2.6.5	IBMDSwitcherKeyChromaParameters::GetYSuppress method	364
5.2.6.6	IBMDSwitcherKeyChromaParameters::SetYSuppress method	364
5.2.6.7	IBMDSwitcherKeyChromaParameters::GetLift method	365
5.2.6.8	IBMDSwitcherKeyChromaParameters::SetLift method	365
5.2.6.9	IBMDSwitcherKeyChromaParameters::GetNarrow method	366
5.2.6.10	IBMDSwitcherKeyChromaParameters::SetNarrow method	366
5.2.6.11	IBMDSwitcherKeyChromaParameters::AddCallback method	367
5.2.6.12	IBMDSwitcherKeyChromaParameters::RemoveCallback method	368
5.2.7	IBMDSwitcherKeyChromaParametersCallback Interface	369
5.2.7.1	IBMDSwitcherKeyChromaParametersCallback::Notify method	370
5.2.8	IBMDSwitcherKeyPatternParameters Interface	371
5.2.8.1	IBMDSwitcherKeyPatternParameters::GetPattern method	373
5.2.8.2	IBMDSwitcherKeyPatternParameters::SetPattern method	374
5.2.8.3	IBMDSwitcherKeyPatternParameters::GetSize method	375
5.2.8.4	IBMDSwitcherKeyPatternParameters::SetSize method	375
5.2.8.5	IBMDSwitcherKeyPatternParameters::GetSymmetry method	376
5.2.8.6	IBMDSwitcherKeyPatternParameters::SetSymmetry method	376
5.2.8.7	IBMDSwitcherKeyPatternParameters::GetSoftness method	377
5.2.8.8	IBMDSwitcherKeyPatternParameters::SetSoftness method	377
5.2.8.9	IBMDSwitcherKeyPatternParameters::GetHorizontalOffset method	378
5.2.8.10	IBMDSwitcherKeyPatternParameters::SetHorizontalOffset method	378
5.2.8.11	IBMDSwitcherKeyPatternParameters::GetVerticalOffset method	379
5.2.8.12	IBMDSwitcherKeyPatternParameters::SetVerticalOffset method	379
5.2.8.13	IBMDSwitcherKeyPatternParameters::GetInverse method	380
5.2.8.14	IBMDSwitcherKeyPatternParameters::SetInverse method	380
5.2.8.15	IBMDSwitcherKeyPatternParameters::AddCallback method	381
5.2.8.16	IBMDSwitcherKeyPatternParameters::RemoveCallback method	382
5.2.9	IBMDSwitcherKeyPatternParametersCallback Interface	383

# Table of Contents

---

5.2.9.1	IBMDSwitcherKeyPatternParametersCallback::Notify method	384
5.2.10	IBMDSwitcherKeyDVEParameters Interface	385
5.2.10.1	IBMDSwitcherKeyDVEParameters::GetShadow method	388
5.2.10.2	IBMDSwitcherKeyDVEParameters::SetShadow method	388
5.2.10.3	IBMDSwitcherKeyDVEParameters::GetLightSourceDirection method	389
5.2.10.4	IBMDSwitcherKeyDVEParameters::SetLightSourceDirection method	389
5.2.10.5	IBMDSwitcherKeyDVEParameters::GetLightSourceAltitude method	390
5.2.10.6	IBMDSwitcherKeyDVEParameters::SetLightSourceAltitude method	390
5.2.10.7	IBMDSwitcherKeyDVEParameters::GetBorderEnabled method	391
5.2.10.8	IBMDSwitcherKeyDVEParameters::SetBorderEnabled method	391
5.2.10.9	IBMDSwitcherKeyDVEParameters::GetBorderBevel method	392
5.2.10.10	IBMDSwitcherKeyDVEParameters::SetBorderBevel method	393
5.2.10.11	IBMDSwitcherKeyDVEParameters::GetBorderWidthIn method	394
5.2.10.12	IBMDSwitcherKeyDVEParameters::SetBorderWidthIn method	394
5.2.10.13	IBMDSwitcherKeyDVEParameters::GetBorderWidthOut method	395
5.2.10.14	IBMDSwitcherKeyDVEParameters::SetBorderWidthOut method	395
5.2.10.15	IBMDSwitcherKeyDVEParameters::GetBorderSoftnessIn method	396
5.2.10.16	IBMDSwitcherKeyDVEParameters::SetBorderSoftnessIn method	396
5.2.10.17	IBMDSwitcherKeyDVEParameters::GetBorderSoftnessOut method	397
5.2.10.18	IBMDSwitcherKeyDVEParameters::SetBorderSoftnessOut method	397
5.2.10.19	IBMDSwitcherKeyDVEParameters::GetBorderBevelSoftness method	398
5.2.10.20	IBMDSwitcherKeyDVEParameters::SetBorderBevelSoftness method	398
5.2.10.21	IBMDSwitcherKeyDVEParameters::GetBorderBevelPosition method	399
5.2.10.22	IBMDSwitcherKeyDVEParameters::SetBorderBevelPosition method	399
5.2.10.23	IBMDSwitcherKeyDVEParameters::GetBorderOpacity method	400
5.2.10.24	IBMDSwitcherKeyDVEParameters::SetBorderOpacity method	400
5.2.10.25	IBMDSwitcherKeyDVEParameters::GetBorderHue method	401
5.2.10.26	IBMDSwitcherKeyDVEParameters::SetBorderHue method	401
5.2.10.27	IBMDSwitcherKeyDVEParameters::GetBorderSaturation method	402
5.2.10.28	IBMDSwitcherKeyDVEParameters::SetBorderSaturation method	402
5.2.10.29	IBMDSwitcherKeyDVEParameters::GetBorderLuma method	403
5.2.10.30	IBMDSwitcherKeyDVEParameters::SetBorderLuma method	403
5.2.10.31	IBMDSwitcherKeyDVEParameters::GetMasked method	404
5.2.10.32	IBMDSwitcherKeyDVEParameters::SetMasked method	404
5.2.10.33	IBMDSwitcherKeyDVEParameters::GetMaskTop method	405
5.2.10.34	IBMDSwitcherKeyDVEParameters::SetMaskTop method	405



# Table of Contents

---

5.2.10.35	IBMDSwitcherKeyDVEParameters::GetMaskBottom method	406
5.2.10.36	IBMDSwitcherKeyDVEParameters::SetMaskBottom method	406
5.2.10.37	IBMDSwitcherKeyDVEParameters::GetMaskLeft method	407
5.2.10.38	BMDSwitcherKeyDVEParameters::SetMaskLeft method	407
5.2.10.39	BMDSwitcherKeyDVEParameters::GetMaskRight method	408
5.2.10.40	BMDSwitcherKeyDVEParameters::SetMaskRight method	408
5.2.10.41	IBMDSwitcherKeyDVEParameters::ResetMask method	409
5.2.10.42	IBMDSwitcherKeyDVEParameters::AddCallback method	410
5.2.10.43	IBMDSwitcherKeyDVEParameters::RemoveCallback method	411
5.2.11	IBMDSwitcherKeyDVEParametersCallback Interface	412
5.2.11.1	IBMDSwitcherKeyDVEParametersCallback::Notify method	413
5.2.12	IBMDSwitcherKeyFlyParameters Interface	414
5.2.12.1	IBMDSwitcherKeyFlyParameters::GetFly method	416
5.2.12.2	IBMDSwitcherKeyFlyParameters::SetFly method	416
5.2.12.3	IBMDSwitcherKeyFlyParameters::GetCanFly method	417
5.2.12.4	IBMDSwitcherKeyFlyParameters::GetRate method	418
5.2.12.5	IBMDSwitcherKeyFlyParameters::SetRate method	419
5.2.12.6	IBMDSwitcherKeyFlyParameters::GetSizeX method	420
5.2.12.7	IBMDSwitcherKeyFlyParameters::SetSizeX method	421
5.2.12.8	IBMDSwitcherKeyFlyParameters::GetSizeY method	422
5.2.12.9	IBMDSwitcherKeyFlyParameters::SetSizeY method	423
5.2.12.10	IBMDSwitcherKeyFlyParameters::GetPositionX method	424
5.2.12.11	IBMDSwitcherKeyFlyParameters::SetPositionX method	425
5.2.12.12	IBMDSwitcherKeyFlyParameters::GetPositionY method	426
5.2.12.13	IBMDSwitcherKeyFlyParameters::SetPositionY method	427
5.2.12.14	IBMDSwitcherKeyFlyParameters::GetRotation method	427
5.2.12.15	IBMDSwitcherKeyFlyParameters::SetRotation method	428
5.2.12.16	IBMDSwitcherKeyFlyParameters::ResetRotation method	428
5.2.12.17	IBMDSwitcherKeyFlyParameters::ResetDVE method	429
5.2.12.18	IBMDSwitcherKeyFlyParameters::ResetDVEFull method	429
5.2.12.19	IBMDSwitcherKeyFlyParameters::IsKeyFrameStored method	430
5.2.12.20	IBMDSwitcherKeyFlyParameters::StoreAsKeyFrame method	431
5.2.12.21	IBMDSwitcherKeyFlyParameters::RunToKeyFrame method	432
5.2.12.22	IBMDSwitcherKeyFlyParameters::IsAtKeyFrames method	433
5.2.12.23	IBMDSwitcherKeyFlyParameters::GetKeyFrameParameters method	434
5.2.12.24	IBMDSwitcherKeyFlyParameters::IsRunning method	435

# Table of Contents

---

5.2.12.25	IBMDSwitcherKeyFlyParameters::AddCallback method	436
5.2.12.26	IBMDSwitcherKeyFlyParameters::RemoveCallback method	437
5.2.13	IBMDSwitcherKeyFlyParametersCallback Interface	438
5.2.13.1	IBMDSwitcherKeyFlyParametersCallback::Notify method	439
5.2.14	IBMDSwitcherKeyFlyKeyFrameParameters Interface	440
5.2.14.1	IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeX method	443
5.2.14.2	IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeX method	443
5.2.14.3	IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeY method	444
5.2.14.4	IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeY method	444
5.2.14.5	IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionX method	445
5.2.14.6	IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionX method	445
5.2.14.7	IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionY method	446
5.2.14.8	IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionY method	446
5.2.14.9	IBMDSwitcherKeyFlyKeyFrameParameters::GetRotation method	447
5.2.14.10	IBMDSwitcherKeyFlyKeyFrameParameters::SetRotation method	447
5.2.14.11	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthOut method	448
5.2.14.12	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthOut method	448
5.2.14.13	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthIn method	449
5.2.14.14	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthIn method	449
5.2.14.15	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessOut method	450
5.2.14.16	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessOut method	450
5.2.14.17	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessIn method	451
5.2.14.18	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessIn method	451
5.2.14.19	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelSoftness method	452
5.2.14.20	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelSoftness method	452
5.2.14.21	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelPosition method	453
5.2.14.22	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelPosition method	453
5.2.14.23	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderOpacity method	454
5.2.14.24	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderOpacity method	454
5.2.14.25	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderHue method	455
5.2.14.26	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderHue method	455
5.2.14.27	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSaturation method	456
5.2.14.28	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSaturation method	456
5.2.14.29	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLuma method	457
5.2.14.30	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLuma method	457
5.2.14.31	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceDirection method	458

# Table of Contents

---

5.2.14.32	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLightSourceDirection method	459
5.2.14.33	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceAltitude method	460
5.2.14.34	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLightSourceAltitude method	460
5.2.14.35	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskTop method	461
5.2.14.36	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskTop method	461
5.2.14.37	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskBottom method	462
5.2.14.38	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskBottom method	462
5.2.14.39	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskLeft method	463
5.2.14.40	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskLeft method	463
5.2.14.41	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskRight method	464
5.2.14.42	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskRight method	464
5.2.14.43	IBMDSwitcherKeyFlyKeyFrameParameters::AddCallback method	465
5.2.14.44	IBMDSwitcherKeyFlyKeyFrameParameters::RemoveCallback method	466
5.2.15	IBMDSwitcherKeyFlyKeyFrameParametersCallback Interface	467
5.2.15.1	IBMDSwitcherKeyFlyKeyFrameParametersCallback::Notify method	468
5.2.16	IBMDSwitcherDownstreamKeyIterator Interface	469
5.2.16.1	IBMDSwitcherDownstreamKeyIterator::Next method	470
5.2.17	IBMDSwitcherDownstreamKey Interface	471
5.2.17.1	IBMDSwitcherDownstreamKey::GetInputCut method	473
5.2.17.2	IBMDSwitcherDownstreamKey::SetInputCut method	474
5.2.17.3	IBMDSwitcherDownstreamKey::GetInputFill method	475
5.2.17.4	IBMDSwitcherDownstreamKey::SetInputFill method	476
5.2.17.5	IBMDSwitcherDownstreamKey::GetFillInputAvailabilityMask method	477
5.2.17.6	IBMDSwitcherDownstreamKey::GetCutInputAvailabilityMask method	478
5.2.17.7	IBMDSwitcherDownstreamKey::GetTie method	479
5.2.17.8	IBMDSwitcherDownstreamKey::SetTie method	479
5.2.17.9	IBMDSwitcherDownstreamKey::GetRate method	480
5.2.17.10	IBMDSwitcherDownstreamKey::SetRate method	481
5.2.17.11	IBMDSwitcherDownstreamKey::GetOnAir method	482
5.2.17.12	IBMDSwitcherDownstreamKey::SetOnAir method	482
5.2.17.13	IBMDSwitcherDownstreamKey::PerformAutoTransition method	483
5.2.17.14	IBMDSwitcherDownstreamKey::IsTransitioning method	483
5.2.17.15	IBMDSwitcherDownstreamKey::IsAutoTransitioning method	484
5.2.17.16	IBMDSwitcherDownstreamKey::GetFramesRemaining method	484
5.2.17.17	IBMDSwitcherDownstreamKey::GetPreMultiplied method	485
5.2.17.18	IBMDSwitcherDownstreamKey::SetPreMultiplied method	486

# Table of Contents

5.2.17.19	IBMDSwitcherDownstreamKey::GetClip method	487
5.2.17.20	IBMDSwitcherDownstreamKey::SetClip method	487
5.2.17.21	IBMDSwitcherDownstreamKey::GetGain method	488
5.2.17.22	IBMDSwitcherDownstreamKey::SetGain method	488
5.2.17.23	IBMDSwitcherDownstreamKey::GetInverse method	489
5.2.17.24	IBMDSwitcherDownstreamKey::SetInverse method	489
5.2.17.25	IBMDSwitcherDownstreamKey::GetMasked method	490
5.2.17.26	IBMDSwitcherDownstreamKey::SetMasked method	490
5.2.17.27	IBMDSwitcherDownstreamKey::GetMaskTop method	491
5.2.17.28	IBMDSwitcherDownstreamKey::SetMaskTop method	491
5.2.17.29	IBMDSwitcherDownstreamKey::GetMaskBottom method	492
5.2.17.30	IBMDSwitcherDownstreamKey::SetMaskBottom method	492
5.2.17.31	IBMDSwitcherDownstreamKey::GetMaskLeft method	493
5.2.17.32	IBMDSwitcherDownstreamKey::SetMaskLeft method	493
5.2.17.33	IBMDSwitcherDownstreamKey::GetMaskRight method	494
5.2.17.34	IBMDSwitcherDownstreamKey::SetMaskRight method	494
5.2.17.35	IBMDSwitcherDownstreamKey::ResetMask method	495
5.2.17.36	IBMDSwitcherDownstreamKey::AddCallback method	496
5.2.17.37	IBMDSwitcherDownstreamKey::RemoveCallback method	497
5.2.18	IBMDSwitcherDownstreamKeyCallback Interface	498
5.2.18.1	IBMDSwitcherDownstreamKeyCallback::Notify	499

## 6 SuperSource 500

6.1	SuperSource Data Types	500
6.1.1	SuperSource Box Event Type	500
6.1.2	SuperSource Input Event Type	501
6.1.3	SuperSource Art Option	502
6.2	Interface Reference	503
6.2.1	IBMDSwitcherInputSuperSource Interface	503
6.2.1.1	IBMDSwitcherInputSuperSource::GetInputCut method	506
6.2.1.2	IBMDSwitcherInputSuperSource::SetInputCut method	507
6.2.1.3	IBMDSwitcherInputSuperSource::GetInputFill method	508
6.2.1.4	IBMDSwitcherInputSuperSource::SetInputFill method	509
6.2.1.5	IBMDSwitcherInputSuperSource::GetFillInputAvailabilityMask method	510
6.2.1.6	IBMDSwitcherInputSuperSource::GetCutInputAvailabilityMask method	511

# Table of Contents

---

6.2.1.7	IBMDSwitcherInputSuperSource::GetArtOption method	512
6.2.1.8	IBMDSwitcherInputSuperSource::SetArtOption method	513
6.2.1.9	IBMDSwitcherInputSuperSource::GetPreMultiplied method	514
6.2.1.10	IBMDSwitcherInputSuperSource::SetPreMultiplied method	515
6.2.1.11	IBMDSwitcherInputSuperSource::GetClip method	516
6.2.1.12	IBMDSwitcherInputSuperSource::SetClip method	516
6.2.1.13	IBMDSwitcherInputSuperSource::GetGain method	517
6.2.1.14	IBMDSwitcherInputSuperSource::SetGain method	517
6.2.1.15	IBMDSwitcherInputSuperSource::GetInverse method	518
6.2.1.16	IBMDSwitcherInputSuperSource::SetInverse method	518
6.2.1.17	IBMDSwitcherInputSuperSource::GetBorderEnabled method	519
6.2.1.18	IBMDSwitcherInputSuperSource::SetBorderEnabled method	519
6.2.1.19	IBMDSwitcherInputSuperSource::GetBorderBevel method	520
6.2.1.20	IBMDSwitcherInputSuperSource::SetBorderBevel method	521
6.2.1.21	IBMDSwitcherInputSuperSource::GetBorderWidthOut method	522
6.2.1.22	IBMDSwitcherInputSuperSource::SetBorderWidthOut method	522
6.2.1.23	IBMDSwitcherInputSuperSource::GetBorderWidthIn method	523
6.2.1.24	IBMDSwitcherInputSuperSource::SetBorderWidthIn method	523
6.2.1.25	IBMDSwitcherInputSuperSource::GetBorderSoftnessOut method	524
6.2.1.26	IBMDSwitcherInputSuperSource::SetBorderSoftnessOut method	524
6.2.1.27	IBMDSwitcherInputSuperSource::GetBorderSoftnessIn method	525
6.2.1.28	IBMDSwitcherInputSuperSource::SetBorderSoftnessIn method	525
6.2.1.29	IBMDSwitcherInputSuperSource::GetBorderBevelSoftness method	526
6.2.1.30	IBMDSwitcherInputSuperSource::SetBorderBevelSoftness method	526
6.2.1.31	IBMDSwitcherInputSuperSource::GetBorderBevelPosition method	527
6.2.1.32	IBMDSwitcherInputSuperSource::SetBorderBevelPosition method	527
6.2.1.33	IBMDSwitcherInputSuperSource::GetBorderHue method	528
6.2.1.34	IBMDSwitcherInputSuperSource::SetBorderHue method	528
6.2.1.35	IBMDSwitcherInputSuperSource::GetBorderSaturation method	529
6.2.1.36	IBMDSwitcherInputSuperSource::SetBorderSaturation method	529
6.2.1.37	IBMDSwitcherInputSuperSource::GetBorderLuma method	530
6.2.1.38	IBMDSwitcherInputSuperSource::SetBorderLuma method	530
6.2.1.39	IBMDSwitcherInputSuperSource::GetBorderLightSourceDirection method	531
6.2.1.40	IBMDSwitcherInputSuperSource::SetBorderLightSourceDirection method	531
6.2.1.41	IBMDSwitcherInputSuperSource::GetBorderLightSourceAltitude method	532
6.2.1.42	IBMDSwitcherInputSuperSource::SetBorderLightSourceAltitude method	532

# Table of Contents

6.2.1.43	IBMDSwitcherInputSuperSource::AddCallback method	533
6.2.1.44	IBMDSwitcherInputSuperSource::RemoveCallback method	534
6.2.2	IBMDSwitcherInputSuperSourceCallback Interface	535
6.2.2.1	IBMDSwitcherInputSuperSourceCallback::Notify method	536
6.2.3	IBMDSwitcherSuperSourceBoxIterator Interface	537
6.2.3.1	IBMDSwitcherSuperSourceBoxIterator::Next method	538
6.2.4	IBMDSwitcherSuperSourceBox Interface	539
6.2.4.1	IBMDSwitcherSuperSourceBox::GetEnabled method	541
6.2.4.2	IBMDSwitcherSuperSourceBox::SetEnabled method	542
6.2.4.3	IBMDSwitcherSuperSourceBox::GetInputSource method	543
6.2.4.4	IBMDSwitcherSuperSourceBox::SetInputSource method	544
6.2.4.5	IBMDSwitcherSuperSourceBox::GetPositionX method	545
6.2.4.6	IBMDSwitcherSuperSourceBox::SetPositionX method	545
6.2.4.7	IBMDSwitcherSuperSourceBox::GetPositionY method	546
6.2.4.8	IBMDSwitcherSuperSourceBox::SetPositionY method	546
6.2.4.9	IBMDSwitcherSuperSourceBox::GetSize method	547
6.2.4.10	IBMDSwitcherSuperSourceBox::SetSize method	547
6.2.4.11	IBMDSwitcherSuperSourceBox::GetCropped method	548
6.2.4.12	IBMDSwitcherSuperSourceBox::SetCropped method	548
6.2.4.13	IBMDSwitcherSuperSourceBox::GetCropTop method	549
6.2.4.14	IBMDSwitcherSuperSourceBox::SetCropTop method	549
6.2.4.15	IBMDSwitcherSuperSourceBox::GetCropBottom method	550
6.2.4.16	IBMDSwitcherSuperSourceBox::SetCropBottom method	550
6.2.4.17	IBMDSwitcherSuperSourceBox::GetCropLeft method	551
6.2.4.18	IBMDSwitcherSuperSourceBox::SetCropLeft method	551
6.2.4.19	IBMDSwitcherSuperSourceBox::GetCropRight method	552
6.2.4.20	IBMDSwitcherSuperSourceBox::SetCropRight method	552
6.2.4.21	IBMDSwitcherSuperSourceBox::ResetCrop method	553
6.2.4.22	IBMDSwitcherSuperSourceBox::GetInputAvailabilityMask method	554
6.2.4.23	IBMDSwitcherSuperSourceBox::AddCallback method	555
6.2.4.24	IBMDSwitcherSuperSourceBox::RemoveCallback method	556
6.2.5	IBMDSwitcherSuperSourceBoxCallback Interface	557
6.2.5.1	IBMDSwitcherSuperSourceBoxCallback::Notify method	558

## 7 Audio Mixing 559

7.1	Audio Mixing Data Types	559
-----	-------------------------	-----

# Table of Contents

---

7.1.1	Audio Mixer Event Type	559
7.1.2	Audio Input Identifier	559
7.1.3	Audio Input Type	559
7.1.4	Audio Mix Option	559
7.1.5	Audio Input Event Type	560
7.1.6	Audio Monitor Output Event Type	560
7.1.7	Switcher Talkback Event Types	560
7.2	Interface Reference	561
7.2.1	IBMDSwitcherAudioMixer Interface	561
7.2.1.1	IBMDSwitcherAudioMixer::GetProgramOutGain method	562
7.2.1.2	IBMDSwitcherAudioMixer::SetProgramOutGain method	562
7.2.1.3	IBMDSwitcherAudioMixer::GetProgramOutBalance method	563
7.2.1.4	IBMDSwitcherAudioMixer::GetProgramOutFollowFadeToBlack method	564
7.2.1.5	IBMDSwitcherAudioMixer::SetProgramOutFollowFadeToBlack method	565
7.2.1.6	IBMDSwitcherAudioMixer::SetProgramOutBalance method	566
7.2.1.7	IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable method	567
7.2.1.8	IBMDSwitcherAudioMixer::ResetProgramOutLevelNotificationPeaks method	568
7.2.1.9	IBMDSwitcherAudioMixer::ResetAllLevelNotificationPeaks method	568
7.2.1.10	IBMDSwitcherAudioMixer::AddCallback method	569
7.2.1.11	IBMDSwitcherAudioMixer::RemoveCallback method	570
7.2.1.12	IBMDSwitcherAudioMixer::CreateIterator method	571
7.2.2	IBMDSwitcherAudioMixerCallback Interface	572
7.2.2.1	IBMDSwitcherAudioMixerCallback::Notify method	573
7.2.2.2	IBMDSwitcherAudioMixerCallback::ProgramOutLevelNotification method	574
7.2.3	IBMDSwitcherAudioInputIterator Interface	575
7.2.3.1	IBMDSwitcherAudioInputIterator::Next method	576
7.2.3.2	IBMDSwitcherAudioInputIterator::GetById method	577
7.2.4	IBMDSwitcherAudioInput Interface	578
7.2.4.1	IBMDSwitcherAudioInput::GetType method	579
7.2.4.2	IBMDSwitcherAudioInput::GetCurrentExternalPortType method	580
7.2.4.3	IBMDSwitcherAudioInput::GetMixOption method	581
7.2.4.4	IBMDSwitcherAudioInput::SetMixOption method	582
7.2.4.5	IBMDSwitcherAudioInput::GetGain method	583
7.2.4.6	IBMDSwitcherAudioInput::SetGain method	583
7.2.4.7	IBMDSwitcherAudioInput::GetBalance method	584
7.2.4.8	IBMDSwitcherAudioInput::SetBalance method	585



# Table of Contents

---

7.2.4.9	IBMDSwitcherAudioInput::IsMixedIn method	586
7.2.4.10	IBMDSwitcherAudioInput::GetAudioInputId method	587
7.2.4.11	IBMDSwitcherAudioInput::ResetLevelNotificationPeaks method	587
7.2.4.12	IBMDSwitcherAudioInput::AddCallback method	588
7.2.4.13	IBMDSwitcherAudioInput::RemoveCallback method	589
7.2.5	IBMDSwitcherAudioInputCallback Interface	590
7.2.5.1	IBMDSwitcherAudioInputCallback::Notify method	591
7.2.5.2	IBMDSwitcherAudioInputCallback::LevelNotification method	592
7.2.6	IBMDSwitcherAudioMonitorOutputIterator Interface	593
7.2.6.1	IBMDSwitcherAudioMonitorOutputIterator::Next method	594
7.2.7	IBMDSwitcherAudioMonitorOutput Interface	595
7.2.7.1	IBMDSwitcherAudioMonitorOutput::GetMonitorEnable method	597
7.2.7.2	IBMDSwitcherAudioMonitorOutput::SetMonitorEnable method	598
7.2.7.3	IBMDSwitcherAudioMonitorOutput::GetMute method	599
7.2.7.4	IBMDSwitcherAudioMonitorOutput::SetMute method	599
7.2.7.5	IBMDSwitcherAudioMonitorOutput::GetGain method	600
7.2.7.6	IBMDSwitcherAudioMonitorOutput::SetGain method	600
7.2.7.7	IBMDSwitcherAudioMonitorOutput::GetSolo method	601
7.2.7.8	IBMDSwitcherAudioMonitorOutput::SetSolo method	601
7.2.7.9	IBMDSwitcherAudioMonitorOutput::GetSoloInput method	602
7.2.7.10	IBMDSwitcherAudioMonitorOutput::SetSoloInput method	603
7.2.7.11	IBMDSwitcherAudioMonitorOutput::GetDim method	604
7.2.7.12	IBMDSwitcherAudioMonitorOutput::SetDim method	604
7.2.7.13	IBMDSwitcherAudioMonitorOutput::GetDimLevel method	605
7.2.7.14	IBMDSwitcherAudioMonitorOutput::SetDimLevel method	605
7.2.7.15	IBMDSwitcherAudioMonitorOutput::ResetLevelNotificationPeaks method	606
7.2.7.16	IBMDSwitcherAudioMonitorOutput::AddCallback method	607
7.2.7.17	IBMDSwitcherAudioMonitorOutput::RemoveCallback method	608
7.2.8	IBMDSwitcherAudioMonitorOutputCallback Interface	609
7.2.8.1	IBMDSwitcherAudioMonitorOutputCallback::Notify method	610
7.2.8.2	IBMDSwitcherAudioMonitorOutputCallback::LevelNotification method	611
7.2.9	IBMDSwitcherTalkback Interface	612
7.2.9.1	IBMDSwitcherTalkback::GetMuteSDI method	613
7.2.9.2	IBMDSwitcherTalkback::SetMuteSDI method	614
7.2.9.3	IBMDSwitcherTalkback::AddCallback method	615
7.2.9.4	IBMDSwitcherTalkback::RemoveCallback method	616

# Table of Contents

7.2.10	IBMDSwitcherTalkbackCallback Interface	617
7.2.10.1	IBMDSwitcherSerialPortCallback::Notify method	618

## 8 Camera Control 619

8.1	Camera Control Data Types	619
8.1.1	Switcher Camera Control Event Type	619
8.1.2	Switcher Camera Control Parameter Type	619
8.2	Interface Reference	620
8.2.1	Switcher Camera Control Parameter Iterator	620
8.2.1.1	IBMDSwitcherCameraControlParameterIterator::Next method	621
8.2.2	IBMDSwitcherCameraControlCallback Interface	622
8.2.2.1	IBMDSwitcherCameraControlCallback::Notify method	623
8.2.3	IBMDSwitcherCameraControl Interface	624
8.2.3.1	IBMDSwitcherCameraControl::CreateIterator method	626
8.2.3.2	IBMDSwitcherCameraControl::GetPeriodicFlushInterval method	627
8.2.3.3	IBMDSwitcherCameraControl::SetPeriodicFlushInterval method	628
8.2.3.4	IBMDSwitcherCameraControl::GetParameterInfo method	629
8.2.3.5	IBMDSwitcherCameraControl::GetParameterPeriodicFlushEnabled method	630
8.2.3.6	IBMDSwitcherCameraControl::SetParameterPeriodicFlushEnabled method	631
8.2.3.7	IBMDSwitcherCameraControl::SetFlags method	632
8.2.3.8	IBMDSwitcherCameraControl::ToggleFlags method	633
8.2.3.9	IBMDSwitcherCameraControl::GetFlags method	634
8.2.3.10	IBMDSwitcherCameraControl::SetBytes method	635
8.2.3.11	IBMDSwitcherCameraControl::OffsetBytes method	636
8.2.3.12	IBMDSwitcherCameraControl::GetBytes method	637
8.2.3.13	IBMDSwitcherCameraControl::SetInt16s method	638
8.2.3.14	IBMDSwitcherCameraControl::OffsetInt16s method	639
8.2.3.15	IBMDSwitcherCameraControl::GetInt16s method	640
8.2.3.16	IBMDSwitcherCameraControl::SetInt32s method	641
8.2.3.17	IBMDSwitcherCameraControl::OffsetInt32s method	642
8.2.3.18	IBMDSwitcherCameraControl::GetInt32s method	643
8.2.3.19	IBMDSwitcherCameraControl::SetInt64s method	644
8.2.3.20	IBMDSwitcherCameraControl::OffsetInt64s method	645
8.2.3.21	IBMDSwitcherCameraControl::GetInt64s method	646
8.2.3.22	IBMDSwitcherCameraControl::OffsetFloats method	647

# Table of Contents

---

8.2.3.23	IBMDSwitcherCameraControl::SetFloats method	648
8.2.3.24	IBMDSwitcherCameraControl::GetFloats method	649
8.2.3.25	IBMDSwitcherCameraControl::AddCallback method	650
8.2.3.26	IBMDSwitcherCameraControl::RemoveCallback method	651

## 9 Macros 652

9.1	General Information	652
9.1.1	Macro Indexes and Identification	652
9.1.2	Recording a Macro	652
9.1.3	Downloading a Macro	653
9.1.4	Uploading a Macro	653
9.1.5	Unsupported Operations	654
9.2	Macro Data Types	655
9.2.1	Macro Pool Event Type	655
9.2.2	Macro Control Event Type	655
9.2.3	Macro Run Status	655
9.2.4	Macro Record Status	655
9.3	Interface Reference	656
9.3.1	IBMDSwitcherMacroPool Interface	656
9.3.1.1	IBMDSwitcherMacroPool::GetMaxCount method	657
9.3.1.2	IBMDSwitcherMacroPool::Delete method	657
9.3.1.3	IBMDSwitcherMacroPool::IsValid method	658
9.3.1.4	IBMDSwitcherMacroPool::HasUnsupportedOps method	659
9.3.1.5	IBMDSwitcherMacroPool::GetName method	660
9.3.1.6	IBMDSwitcherMacroPool::SetName method	661
9.3.1.7	IBMDSwitcherMacroPool::GetDescription method	662
9.3.1.8	IBMDSwitcherMacroPool::SetDescription method	663
9.3.1.9	IBMDSwitcherMacroPool::CreateMacro method	664
9.3.1.10	IBMDSwitcherMacroPool::Upload method	665
9.3.1.11	IBMDSwitcherMacroPool::Download method	666
9.3.1.12	IBMDSwitcherMacroPool::AddCallback method	667
9.3.1.13	IBMDSwitcherMacroPool::RemoveCallback method	668
9.3.2	IBMDSwitcherTransferMacro Interface	669
9.3.2.1	IBMDSwitcherTransferMacro::Cancel method	670
9.3.2.2	IBMDSwitcherTransferMacro::GetProgress method	670

# Table of Contents

---

9.3.2.3	IBMDSwitcherTransferMacro::GetMacro method	671
9.3.3	IBMDSwitcherMacro Interface	672
9.3.3.1	IBMDSwitcherMacro::GetSize method	673
9.3.3.2	IBMDSwitcherMacro::GetBytes method	673
9.3.4	IBMDSwitcherMacroPoolCallback Interface	674
9.3.4.1	IBMDSwitcherMacroPoolCallback::Notify method	675
9.3.5	IBMDSwitcherMacroControl Interface	676
9.3.5.1	IBMDSwitcherMacroControl::Run method	677
9.3.5.2	IBMDSwitcherMacroControl::GetLoop method	678
9.3.5.3	IBMDSwitcherMacroControl::SetLoop method	679
9.3.5.4	IBMDSwitcherMacroControl::ResumeRunning method	680
9.3.5.5	IBMDSwitcherMacroControl::StopRunning method	680
9.3.5.6	IBMDSwitcherMacroControl::Record method	681
9.3.5.7	IBMDSwitcherMacroControl::RecordUserWait method	682
9.3.5.8	IBMDSwitcherMacroControl::RecordPause method	682
9.3.5.9	IBMDSwitcherMacroControl::StopRecording method	683
9.3.5.10	IBMDSwitcherMacroControl::GetRunStatus method	684
9.3.5.11	IBMDSwitcherMacroControl::GetRecordStatus method	685
9.3.5.12	IBMDSwitcherMacroControl::AddCallback method	686
9.3.5.13	IBMDSwitcherMacroControl::RemoveCallback method	687
9.3.6	IBMDSwitcherMacroControlCallback Interface	688
9.3.6.1	IBMDSwitcherMacroControlCallback::Notify method	689

# SECTION 1 Introduction

---

## 1.1 Welcome

Thanks for downloading the Blackmagic Design Switcher Software Developers Kit (SDK).

## 1.2 Overview

The Switcher SDK provides a stable, cross- platform interface to a Blackmagic Design Switcher product line. The SDK provides both low-level control of hardware and high-level interfaces to allow developers to easily perform common tasks.

The Switcher SDK consists of a set of interface descriptions & sample applications which demonstrate the use of the features of the Switcher hardware. The details of the SDK are described in this document. Some Switcher capabilities, such as uncompressed USB 3 video capture and H.264 video streaming, must be accessed using the separate DeckLink SDK.

The Switcher SDK supports Microsoft Windows and Mac OS X.

You can download the Switcher SDK and DeckLink SDK from the Blackmagic Design support center at: **[www.blackmagicdesign.com/support](http://www.blackmagicdesign.com/support)**  
The product family is ATEM Live production switchers.

If you're looking for detailed answers regarding technologies used by Blackmagic Design, such as codecs, core media, APIs, SDK and more, visit the Blackmagic Software Developers Forum. The forum is a helpful place for you to engage with both Blackmagic support staff and other forum members who can answer developer specific questions and provide further information. The Software Developers forum can be found within the Blackmagic Design Forum at **[forum.blackmagicdesign.com](http://forum.blackmagicdesign.com)**

If you wish to ask questions outside of the software developers forum, please contact us at: **[developer@blackmagicdesign.com](mailto:developer@blackmagicdesign.com)**

## 1.3 API Design

### 1.3.1 Overview

The libraries supporting the Blackmagic SDK are shipped as part of the product installers for each supported product line. Applications built against the interfaces shipped in the SDK will dynamically link against the library installed on the end-user's system.

### 1.3.2 Object Model

The SDK interface is modeled on Microsoft's Component Object Model (COM). On Microsoft Windows platforms, it is provided as a native COM interface registered with the operating system. On other platforms application code is provided to allow the same COM style interface to be used.

The COM model provides a paradigm for creating flexible and extensible interfaces without imposing much overhead or baggage.

### 1.3.3 Object Interfaces

The Switcher API provides programmatic access to all current Blackmagic Design ATEM Switchers.

The Switcher API provides high-level interfaces for configuring switcher inputs and performing switcher operations such as making a transition. Some switcher devices delivering uncompressed video over USB 3 or H.264 video streams over USB 2, but this capability must be accessed using the DeckLink API. Refer to the documentation for the DeckLink SDK at **[www.blackmagicdesign.com/support](http://www.blackmagicdesign.com/support)**.

Functionality within the API is accessed via "object interfaces". Each object in the system may inherit from and be accessed via a number of object interfaces. Typically the developer is able to interact with object interfaces and leave the underlying objects to manage themselves.

Each object interface class has a Globally Unique ID (GUID) called an "Interface ID". On platforms with native COM support, an IID may be used to obtain a handle to an exported interface object from the OS, which is effectively an entry point to an installed API.

Each interface may have related interfaces that are accessed by providing an IID to an existing object interface (see `IUnknown::QueryInterface`). This mechanism allows new interfaces to be added to the API without breaking API or ABI compatibility.

### 1.3.4 Reference Counting

The API uses reference counting to manage the life cycle of object interfaces. The developer may need to add or remove references on object interfaces (**see IUnknown::AddRef and IUnknown::Release**) to influence their life cycle as appropriate in the application.

### 1.3.5 Interface Stability

The SDK provides a set of stable interfaces for accessing Blackmagic Design hardware. Whilst the published interfaces will remain stable, developers need to be aware of some issues they may encounter as new products, features and interfaces become available.

#### 1.3.5.1 New Interfaces

Major pieces of new functionality may be added to the SDK as a whole new object interface. Already released applications will not be affected by the additional functionality. Developers making use of the new functionality should be sure to check the return of **CoCreateInstance** and/or **QueryInterface** as these interfaces will not be available on users systems which are running an older release of the Blackmagic software.

Developers can choose to either reduce the functionality of their application when an interface is not available, or to notify the user that they must install a later version of the Blackmagic software.



#### 1.3.5.2 Updated Interfaces

As new functionality is added to the SDK, some existing interfaces may need to be modified or extended. To maintain compatibility with released software, the original interface will be deprecated but will remain available and maintain its unique identifier (IID). The replacement interface will have a new identifier and remain as similar to the original as possible.

#### 1.3.5.3 Deprecated Interfaces

Interfaces which have been replaced with an updated version, or are no longer recommended for use are “deprecated”. Deprecated interfaces are moved out of the main interface description files into an interface description file named according to the release in which the interface was deprecated. Deprecated interfaces are also renamed with a suffix indicating the release prior to the one in which they were deprecated.

It is recommended that developers update their applications to use the most recent SDK interfaces when they release a new version of their applications. As an interim measure, developers may include the deprecated interface descriptions, and updating the names of the interfaces in their application to access the original interface functionality.

#### 1.3.5.4 Removed interfaces

Interfaces that have been deprecated for some time may eventually be removed in a major driver update if they become impractical to support.

## 1.4 Interface Reference

Every object interface subclasses the **IUnknown** interface.

### 1.4.1 IUnknown Interface

Each API interface is a subclass of the standard COM base class – **IUnknown**. The **IUnknown** object interface provides reference counting and the ability to look up related interfaces by interface ID. The interface ID mechanism allows interfaces to be added to the API without impacting existing applications.

Public Member Functions	
Method	Description
QueryInterface	Provides access to supported child interfaces of the object.
AddRef	Increments the reference count of the object.
Release	Decrements the reference count of the object. When the final reference is removed, the object is freed.

#### 1.4.1.1 IUnknown::QueryInterface method

The **QueryInterface** method looks up a related interface of an object interface.

##### Syntax

```
HRESULT QueryInterface(REFIID id, void **outputInterface);
```

##### Parameters

Name	Direction	Description
id	in	Interface ID of interface to lookup.
output Interface	out	New object interface or NULL on failure.

##### Return Values

Value	Description
E_NOINTERFACE	Interface was not found.
S_OK	Success.

#### 1.4.1.2 IUnknown::AddRef method

The **AddRef** method increments the reference count for an object interface.

##### Syntax

**ULONG**           AddRef();

##### Return Values

Value	Description
Count	New reference count – for debug purposes only.

#### 1.4.1.3 IUnknown::Release method

The Release method decrements the reference count for an object interface.  
When the last reference is removed from an object, the object will be destroyed.

##### Syntax

**ULONG**           Release();

##### Return Values

Value	Description
Count	New reference count – for debug purposes only.

# SECTION 1 API Design

---

## 1.5 Using the Switcher API in a project

The supplied sample applications provide examples of how to include the Switcher API in a project on each supported platform.

To use the Switcher API in your project, one or more files need to be included:

Windows            Switchers X.Y\Win\Include\BMDSwitcherAPI.idl

Mac OS X           Switchers X.Y/Mac/Include/BMDSwitcherAPI.h

Switchers X.Y/Mac/Include/BMDSwitcherAPIDispatch.cpp

### 1.5.1 Basic Types

#### **boolean**

boolean is represented differently on each platform by using its system type:

Windows	BOOL
Mac OS X	bool

#### **string**

Strings are represented differently on each platform, using the most appropriate system type:

Windows	BSTR
Mac OS X	CFStringRef

#### **int64\_t**

The signed 64 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	long long
Mac OS X	int64_t

#### **int32\_t**

The signed 32 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	int
Mac OS X	int32_t

### **uint32\_t**

The unsigned 32 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	unsigned int
Mac OS X	uint32_t

### **int16\_t**

The signed 16 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	short
Mac OS X	int16_t

## **1.5.2 Accessing Switcher devices**

Switcher devices are accessed via the **IBMDSwitcherDiscovery** interface. How a reference to an **IBMDSwitcherDiscovery** is obtained varies between platforms depending on their level of support for COM:

### **1.5.2.1 Windows**

The main entry point to the Switcher API is the **CBMDSwitcherDiscovery** class. This class should be obtained from COM using CoCreateInstance:

```
BMDSwitcherDiscovery* switcherDiscovery = NULL;

CoCreateInstance(CLSID_CBMDSwitcherDiscovery, NULL, CLSCTX_ALL, IID_IBMDSwitcherDiscovery,
                (void**)&switcherDiscovery));
```

On success, CoCreateInstance returns an HRESULT of S\_OK and switcherDiscovery points to a new **IBMDSwitcherDiscovery** object interface.



#### 1.5.2.2 Mac OS X

On Mac OS X a C++ entry point is provided to access an **IBMDSwitcherDiscovery** interface:

```
IBMDSwitcherDiscovery* switcherDiscovery = CreateBMDSwitcherDiscoveryInstance();
```

On success, switcherDiscovery will point to a new **IBMDSwitcherDiscovery** object interface otherwise it will be set to NULL.

## SECTION 2 Basic Switcher Control

### 2 Basic Switcher Control

The Switcher API provides a framework for controlling ATEM switcher devices. The API enables operations such as configuring switcher inputs, performing a transition and making a cut.

#### 2.1 General Information

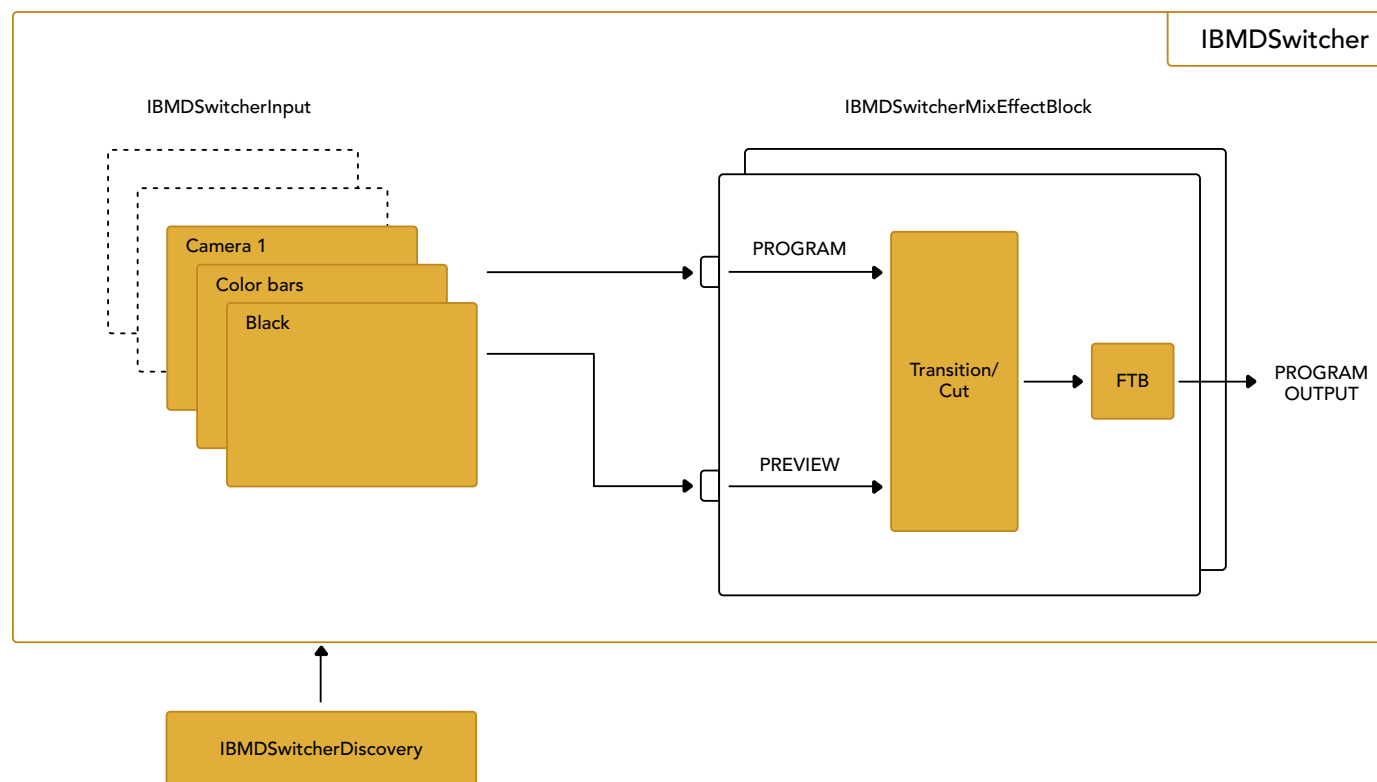
##### 2.1.1 Switcher Configuration and Transitions

An application for controlling a switcher may perform the following steps:

- Use **IBMDSwitcherDiscovery::ConnectTo** to connect to a switcher device and obtain an **IBMDSwitcher** object interface
- Use **IBMDSwitcher::CreateIterator** to get an **IBMDSwitcherInputIterator** object interface
- For each **IBMDSwitcherInput** returned by **IBMDSwitcherInputIterator::Next** retrieve the input's unique Id using **IBMDSwitcherInput::GetInputId** and retrieve other properties of the input, such as the input's name, using **IBMDSwitcherInput::GetString** or **IBMDSwitcherInput::GetInt**
- Use **IBMDSwitcher::CreateIterator** to get an **IBMDSwitcherMixEffectBlockIterator** object interface
- Obtain an **IBMDSwitcherMixEffectBlock** object interface using **IBMDSwitcherMixEffectBlockIterator::Next**
- Use **IBMDSwitcherMixEffectBlock::SetInt** to configure the Program and Preview inputs to the mix effect block by assigning the input Ids returned by **IBMDSwitcherInput::GetInputId**
- Perform a transition between Program and Preview inputs by calling **IBMDSwitcherMixEffectBlock::PerformTransition**
- Install a callback using **IBMDSwitcherMixEffectBlock::AddCallback** and receive **IBMDSwitcherMixEffectBlockCallback::PropertyChanged** callbacks to determine progress through the transition and when the transition is complete

## SECTION 2 Basic Switcher Control

### 2.1.2 Switcher Interface Diagram



# SECTION 2 Basic Switcher Control

## 2.2 Switcher Data Types

### 2.2.1 Basic Switcher Data Types

#### **BMDSwitcherInputId**

BMDSwitcherInputId is a signed 64 bit integer type and used as a unique Id for each switcher input.

### 2.2.2 Switcher Event Type

**BMDSwitcherEventType** enumerates the possible event types for **IBMDSwitcher**.

#### **bmdSwitcherEventTypeVideoModeChanged**

The video standard changed.

#### **bmdSwitcherEventTypeMethodForDownConvertedSDChanged**

The method for down converted SD output has changed.

#### **bmdSwitcherEventTypeDownConvertedHDVideoModeChanged**

The down converted HD output video standard changed for a particular core video standard.

#### **bmdSwitcherEventTypeMultiViewVideoModeChanged**

The MultiView standard changed for a particular core video standard.

#### **bmdSwitcherEventTypePowerStatusChanged**

The power status changed.

#### **bmdSwitcherEventTypeDisconnected**

The switcher disconnected.

### 2.2.3 Switcher Power Status

**BMDSwitcherPowerStatus** enumerates the possible power status flags. This type is used by the **IBMDSwitcher** object interface.

#### **bmdSwitcherPowerStatusSupply1**

Supply 1 has power.

#### **bmdSwitcherPowerStatusSupply2**

**Supply 2 has power.**

## SECTION 2 Basic Switcher Control

### 2.2.4 Switcher Video Mode

**BMDSwitcherVideoMode** enumerates the video standards employed by the switcher.

<b>bmdSwitcherVideoMode525i5994NTSC</b>	525 pixels high, interlaced at 59.94Hz (NTSC).
<b>bmdSwitcherVideoMode625i50PAL</b>	625 pixels high, interlaced at 50Hz (PAL).
<b>bmdSwitcherVideoMode525i5994Anamorphic</b>	525 pixels high, interlaced at 59.94Hz (anamorphic 16:9 widescreen).
<b>bmdSwitcherVideoMode625i50Anamorphic</b>	625 pixels high, interlaced at 50Hz (anamorphic 16:9 widescreen).
<b>bmdSwitcherVideoMode720p50</b>	720 pixels high, progressively scanned at 50Hz.
<b>bmdSwitcherVideoMode720p5994</b>	720 pixels high, progressively scanned at 59.94Hz.
<b>bmdSwitcherVideoMode1080i50</b>	1080 pixels high, interlaced at 50Hz.
<b>bmdSwitcherVideoMode1080i5994</b>	1080 pixels high, interlaced at 59.94Hz.
<b>bmdSwitcherVideoMode1080p2398</b>	1080 pixels high, progressively scanned at 23.98Hz.
<b>bmdSwitcherVideoMode1080p24</b>	1080 pixels high, progressively scanned at 24Hz.
<b>bmdSwitcherVideoMode1080p25</b>	1080 pixels high, progressively scanned at 25Hz.
<b>bmdSwitcherVideoMode1080p2997</b>	1080 pixels high, progressively scanned at 29.97Hz.
<b>bmdSwitcherVideoMode1080p50</b>	1080 pixels high, progressively scanned at 50Hz.
<b>bmdSwitcherVideoMode1080p5994</b>	1080 pixels high, progressively scanned at 59.94Hz.
<b>bmdSwitcherVideoMode4KHDp2398</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 23.98Hz.
<b>bmdSwitcherVideoMode4KHDp24</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 24Hz.
<b>bmdSwitcherVideoMode4KHDp25</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 25Hz.
<b>bmdSwitcherVideoMode4KHDp2997</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 29.97Hz.
<b>bmdSwitcherVideoMode4KHDp50</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 50Hz.
<b>bmdSwitcherVideoMode4KHDp5994</b>	2160 pixels high, 3840 pixels wide, progressively scanned at 59.94Hz.

## SECTION 2 Basic Switcher Control

### 2.2.5 Switcher Down Conversion Methods

**BMDSwitcherDownConversionMethod** enumerates the possible methods for SD down conversion between broadcast standards.

<b>bmdSwitcherDownConversionMethodCentreCut</b>	Centre cut conversion.
<b>bmdSwitcherDownConversionMethodLetterbox</b>	Letter box conversion.
<b>bmdSwitcherDownConversionMethodAnamorphic</b>	Anamorphic conversion.

### 2.2.6 Switcher Input Event Types

**BMDSwitcherInputEventType** enumerates the possible event types for a switcher input object.

<b>bmdSwitcherInputEventTypeShortNameChanged</b>	The short name of the input changed.
<b>bmdSwitcherInputEventTypeLongNameChanged</b>	The long name of the input changed.
<b>bmdSwitcherInputEventTypesProgramTalliedChanged</b>	Program tallying for this input was turned on or turned off.
<b>bmdSwitcherInputEventTypesPreviewTalliedChanged</b>	Preview tallying for this input was turned on or turned off.
<b>bmdSwitcherInputEventTypeAvailableExternalPortTypesChanged</b>	The external port types available to this input changed.
<b>bmdSwitcherInputEventTypeCurrentExternalPortTypeChanged</b>	The current external port type of this input changed.

## SECTION 2 Basic Switcher Control

### 2.2.7 Switcher External Port Types

**BMDSwitcherExternalPortType** enumerates the different kinds of external port type for an input.

This enumeration represents a bitset since an input port may support more than one connection type.

<b>bmdSwitcherExternalPortTypeSDI</b>	SDI connection.
<b>bmdSwitcherExternalPortTypeHDMI</b>	HDMI connection.
<b>bmdSwitcherExternalPortTypeComponent</b>	Component connection.
<b>bmdSwitcherExternalPortTypeComposite</b>	Composite connection, only available in SD video modes.
<b>bmdSwitcherExternalPortTypeSVideo</b>	SVideo connection, only available in SD video modes.
<b>bmdSwitcherExternalPortTypeXLR</b>	XLR audio connection.
<b>bmdSwitcherExternalPortTypeAESEBU</b>	AES EBU audio connection.
<b>bmdSwitcherExternalPortTypeRCA</b>	RCA audio connection.
<b>bmdSwitcherExternalPortTypeInternal</b>	This is an internal input port.

### 2.2.8 Switcher Port Types

**BMDSwitcherPortType** enumerates the possible switcher input port types available for switching.

<b>bmdSwitcherPortTypeExternal</b>	The port is an external port with a physical connector.
<b>bmdSwitcherPortTypeBlack</b>	The port is a black video generator port.
<b>bmdSwitcherPortTypeColorBars</b>	The port is a colorbars generator port.
<b>bmdSwitcherPortTypeColorGenerator</b>	The port is a color generator port.
<b>bmdSwitcherPortTypeMediaPlayerFill</b>	The port is a media player fill port.
<b>bmdSwitcherPortTypeMediaPlayerCut</b>	The port is a media player cut port.
<b>bmdSwitcherPortTypeSuperSource</b>	The port is a super source port.
<b>bmdSwitcherPortTypeMixEffectBlockOutput</b>	The port is a mix effect block output port.
<b>bmdSwitcherPortTypeAuxOutput</b>	The port is an auxiliary output port.

## SECTION 2 Basic Switcher Control

### 2.2.9 Switcher Input Availability

**BMDSwitcherInputAvailability** enumerates the different kinds of input availability for a port.

This enumeration represents a bitset since an input can have multiple availabilities.

**bmdSwitcherInputAvailabilityMixEffectBlock0**

The input is available to be used by mix effect block 0.

**bmdSwitcherInputAvailabilityMixEffectBlock1**

The input is available to be used by mix effect block 1.

**bmdSwitcherInputAvailabilityAuxOutputs**

The input is available to be used by aux outputs.

**bmdSwitcherInputAvailabilityMultiView**

The input can be routed to a MultiView window.

**bmdSwitcherInputAvailabilitySuperSourceArt**

The input is available to be used for SuperSource Art.

**bmdSwitcherInputAvailabilitySuperSourceBox**

The input is available to be used within a SuperSource Box.

**bmdSwitcherInputAvailabilityInputCut**

The input is available to be used as a cut.

### 2.2.10 Switcher Mix Effect Block Properties

**BMDSwitcherMixEffectBlockPropertyId** enumerates the possible properties of a Mix Effect Block.

**bmdSwitcherMixEffectBlockPropertyIdProgramInput**

The program input as a **BMDSwitcherInputId**.

**bmdSwitcherMixEffectBlockPropertyIdPreviewInput**

The preview input as a **BMDSwitcherInputId**.

**bmdSwitcherMixEffectBlockPropertyIdTransitionPosition**

Float value between 0 and 1 representing the progress through a transition where 0 is the start of a transition and 1 is the end.

**bmdSwitcherMixEffectBlockPropertyIdTransitionFramesRemaining**

Number of frames remaining in a transition as an integer.

**bmdSwitcherMixEffectBlockPropertyIdInTransition**

State of mix effect block being in a transition as a boolean.

**bmdSwitcherMixEffectBlockPropertyIdFadeToBlackFramesRemaining**

Number of frames remaining in a fade to black as an integer.

**bmdSwitcherMixEffectBlockPropertyIdInFadeToBlack**

State of mix effect block being in a fade to black as a boolean.

**bmdSwitcherMixEffectBlockPropertyIdPreviewLive**

Boolean value which is true when preview is live.

**bmdSwitcherMixEffectBlockPropertyIdPreviewTransition**

Boolean value which is true when preview transition is active.

**bmdSwitcherMixEffectBlockPropertyIdInputAvailabilityMask**

The corresponding **BMDSwitcherInputAvailability** bit value for this mix effect block.

**bmdSwitcherMixEffectBlockPropertyIdFadeToBlackRate**

The fade to black rate in frames



## SECTION 2 Basic Switcher Control

**bmdSwitcherMixEffectBlockPropertyIdFadeToBlackFullyBlack**  
**bmdSwitcherMixEffectBlockPropertyIdFadeToBlackInTransition**

Boolean value which is true when fade to black is fully black  
Boolean value which is true when fade to black is transitioning.

### 2.2.11 Switcher Connection Errors

**BMDSwitcherConnectToFailure** enumerates the possible errors that can occur when connecting to a switcher.

**bmdSwitcherConnectToFailureNoResponse**  
**bmdSwitcherConnectToFailureIncompatibleFirmware**

The Switcher did not respond after a connection attempt was made.  
The software on the Switcher is incompatible with the current version of the Switcher SDK.

**bmdSwitcherConnectToFailureCorruptData**  
**bmdSwitcherConnectToFailureStateSync**  
**bmdSwitcherConnectToFailureStateSyncTimedOut**

Corrupt data was received during a connection attempt.  
State synchronisation failed during a connection attempt.  
State synchronisation timed out during a connection attempt.

### 2.2.12 Switcher MultiView Layouts

**BMDSwitcherMultiViewLayout** enumerates the possible layout formats for MultiView.

**bmdSwitcherMultiViewLayoutProgramTop**  
**bmdSwitcherMultiViewLayoutProgramBottom**  
**bmdSwitcherMultiViewLayoutProgramLeft**  
**bmdSwitcherMultiViewLayoutProgramRight**

Program and preview reside in upper section of the screen, windows reside below.  
Program and preview reside in lower section of the screen, windows reside above.  
Program and preview reside on left hand side, windows reside on the right.  
Program and preview reside on right hand side, windows reside on the left.

### 2.2.13 Switcher Serial Port Functions

**BMDSwitcherSerialPortFunction** enumerates the functions the serial port can perform.

**bmdSwitcherSerialPortFunctionNone**  
**bmdSwitcherSerialPortFunctionPtzVisca**  
**bmdSwitcherSerialPortFunctionGvg100**

The serial port is not being used.  
The serial port is used to control a Pan-Tilt-Zoom camera using the VISCA protocol.  
If the serial port is set to this function, the switcher can be controlled by a connected GVG100 compatible editor/controller.

## SECTION 2 Basic Switcher Control

### 2.2.14 Switcher Color Events

**BMDSwitcherInputColorEventType** enumerates the possible event types for **IBMDSwitcherInputColorCallback**.

<b>bmdSwitcherInputColorEventTypeHueChanged</b>	The hue changed.
<b>bmdSwitcherInputColorEventTypeSaturationChanged</b>	The saturation changed.
<b>bmdSwitcherInputColorEventTypeLumaChanged</b>	The luma changed.

### 2.2.15 Switcher Aux Events

**BMDSwitcherInputAuxEventType** enumerates the possible event types for **IBMDSwitcherInputAuxCallback**.

<b>bmdSwitcherInputAuxEventTypeInputSourceChanged</b>	The input source changed.
---	---------------------------

### 2.2.16 Switcher MultiView Events

**BMDSwitcherMultiViewEventType** enumerates the possible event types for **IBMDSwitcherMultiViewCallback**.

<b>bmdSwitcherMultiViewEventTypeLayoutChanged</b>	The layout changed.
<b>bmdSwitcherMultiViewEventTypeWindowChanged</b>	Routing to a MultiView window has changed.

### 2.2.17 Switcher Serial Port Event Types

**BMDSwitcherSerialPortEventType** enumerates the possible event types for **IBMDSwitcherSerialPortCallback**.

<b>bmdSwitcherSerialPortEventTypeFunctionChanged</b>	The function of the serial port has changed.
--	--

## SECTION 2 Basic Switcher Control

### 2.3 Interface Reference

#### 2.3.1 IBMDSwitcherDiscovery Interface

The **IBMDSwitcherDiscovery** object interface is used to connect to a physical switcher device.

A reference to an IBMDSwitcherDiscovery object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateBMDSwitcherDiscoveryInstance** on other platforms.

#### Public Member Functions

Method	Description
ConnectTo	Connect to a switcher

## SECTION 2 Basic Switcher Control

### 2.3.1.1 IBMDSwitcherDiscovery::ConnectTo method

The ConnectTo method connects to the specified switcher and returns an **IBMDSwitcher** object interface for the switcher.

**Note:** **ConnectTo** performs a synchronous network connection. This may take several seconds depending upon hostname resolution and network response times.

#### Syntax

```
HRESULT ConnectTo (string deviceAddress, IBMDSwitcher** switcherDevice,  
BMDSwitcherConnectToFailure* failReason);
```

#### Parameters

Name	Direction	Description
deviceAddress	in	Network hostname or IP address of switcher to connect to.
switcherDevice	out	<b>IBMDSwitcher</b> object interface for the connected switcher.
failReason	out	Reason for connection failure as a <b>BMDSwitcherConnectToFailure</b> value.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The switcherDevice or failReason parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2 IBMDSwitcher Interface

The **IBMDSwitcher** object interface represents a physical switcher device.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherDiscovery	IID_IBMDSwitcherDiscovery	An <b>IBMDSwitcher</b> object will be returned after a successful call to the <b>IBMDSwitcherDiscovery::ConnectTo</b> method

#### Public Member Functions

Method	Description
GetProductName	Get the product name of the switcher.
GetVideoMode	Get the current video standard applied across the switcher.
SetVideoMode	Set the video standard applied across the switcher.
DoesSupportVideoMode	Determines if a video standard is supported by the switcher.
GetMethodForDownConvertedSD	Get the SD conversion method applied when down converting between broadcast standards.
SetMethodForDownConvertedSD	Set the SD conversion method applied when down converting between broadcast standards.
GetDownConvertedHDVideoMode	Get the down converted HD video standard for a particular core video standard.
SetDownConvertedHDVideoMode	Set the down converted HD video standard for a particular core video standard.
DoesSupportDownConvertedHDVideoMode	Determines if a down converted HD video standard is supported by a particular core video standard.
GetMultiViewVideoMode	Get the MultiView video standard for a particular core video standard.
SetMultiViewVideoMode	Set the MultiView video standard for a particular core video standard.
DoesSupportMultiViewVideoMode	Determines if a MultiView video standard is supported by a particular core video standard.
GetPowerStatus	Gets the power status of the switcher.
CreateIterator	Create an iterator.
AddCallback	Add a callback to receive switcher events.
RemoveCallback	Remove a callback.

## SECTION 2 Basic Switcher Control

### 2.3.2.1 IBMDSwitcher:: GetProductName method

The **GetProductName** method gets the product name of the switcher.

#### Syntax

```
HRESULT      GetProductName (string* productName);
```

#### Parameters

Name	Direction	Description
productName	out	The product name of the switcher.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The productName parameter is invalid.

### 2.3.2.2 IBMDSwitcher:: GetVideoMode method

The **GetVideoMode** method gets the current video standard applied across the switcher.

#### Syntax

```
HRESULT      GetVideoMode (BMDSwitcherVideoMode* videoMode);
```

#### Parameters

Name	Direction	Description
videoMode	out	The current video standard applied across the switcher.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The videoMode parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.3 IBMDSwitcher:: SetVideoMode method

The **SetVideoMode** method sets the video standard applied across the switcher.

#### Syntax

```
HRESULT SetVideoMode (BMDSwitcherVideoMode videoMode);
```

#### Parameters

Name	Direction	Description
videoMode	in	The video standard applied across the switcher.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The videoMode parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.4 IBMDSwitcher:: DoesSupportVideoMode method

The **DoesSupportVideoMode** method determines if a video standard is supported by the switcher.

#### Syntax

```
HRESULT DoesSupportVideoMode (BMDSwitcherVideoMode videoMode, boolean* supported);
```

#### Parameters

Name	Direction	Description
videoMode	in	The video standard.
supported	out	Boolean value that is true if the video standard is supported.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The supported parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.2.5 IBMDSwitcher:: GetMethodForDownConvertedSD method

The **GetMethodForDownConvertedSD** method gets the SD conversion method applied when down converting between broadcast standards.

#### Syntax

```
HRESULT GetMethodForDownConvertedSD (BMDSwitcherDownConversionMethod* method);
```

#### Parameters

Name	Direction	Description
method	out	The conversion method.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The method parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.6 IBMDSwitcher:: SetMethodForDownConvertedSD method

The **SetMethodForDownConvertedSD** method sets the SD conversion method applied when down converting between broadcast standards.

#### Syntax

```
HRESULT SetMethodForDownConvertedSD (BMDSwitcherDownConversionMethod method);
```

#### Parameters

Name	Direction	Description
method	in	The conversion method.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The method parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.7 IBMDSwitcher:: GetDownConvertedHDVideoMode method

The **GetDownConvertedHDVideoMode** method gets the down converted HD output video standard for a particular core video standard.

#### Syntax

```
HRESULT GetDownConvertedHDVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode* downConvertedHDVideoMode);
```

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard to be down converted.
downConvertedHDVideoMode	out	The mode to which the core video standard is down converted.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The downConvertedHDVideoMode parameter is invalid.
E_INVALIDARG	The coreVideoMode parameter is invalid or not supported.
E_NOTIMPL	HD down conversion is not supported.

## SECTION 2 Basic Switcher Control

### 2.3.2.8 IBMDSwitcher:: SetDownConvertedHDVideoMode method

The **SetDownConvertedHDVideoMode** method sets the down converted HD output video standard for a particular core video standard.

#### Syntax

```
HRESULT SetDownConvertedHDVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode  
downConvertedHDVideoMode);
```

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard to be down converted.
downConvertedHDVideoMode	in	The mode to which the core video standard is to be down converted.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The coreVideoMode or downConvertedHDVideoMode parameter is invalid or not supported.
E_NOTIMPL	HD down conversion is not supported.

## SECTION 2 Basic Switcher Control

### 2.3.2.9 IBMDSwitcher:: DoesSupportDownConvertedHDVideoMode method

The **DoesSupportDownConvertedHDVideoMode** method determines if a down converted HD output video standard is supported by a particular core video standard.

#### Syntax

**HRESULT** DoesSupportDownConvertedHDVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode downConvertedHDVideoMode, boolean\* supported);

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard to be down converted.
downConvertedHDVideoMode	in	The down converted video standard to determine support for.
supported	out	Boolean value that is true if the downConvertedHDVideoMode is supported for the core video standard.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The supported parameter is invalid.
E_INVALIDARG	The downConvertedHDVideoMode parameter is invalid.
E_NOTIMPL	The switcher does not support HD down conversion.

## SECTION 2 Basic Switcher Control

### 2.3.2.10 IBMDSwitcher:: GetMultiViewVideoMode method

The **GetMultiViewVideoMode** method gets the MultiView video standard for a particular core video standard.

#### Syntax

```
HRESULT GetMultiViewVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode* multiviewVideoMode);
```

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard.
multiviewVideoMode	out	The MultiView standard used with the core video standard.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The multiviewVideoMode parameter is invalid.
E_INVALIDARG	The coreVideoMode parameter is invalid or not supported.

## SECTION 2 Basic Switcher Control

### 2.3.2.11 IBMDSwitcher:: SetMultiViewVideoMode method

The **SetMultiViewVideoMode** method gets the MultiView video standard for a particular core video standard.

#### Syntax

```
HRESULT SetMultiViewVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode multiviewVideoMode);
```

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard.
multiviewVideoMode	in	The MultiView standard to set with the core video standard.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The coreVideoMode or multiviewVideoMode parameter is invalid or not supported.

## SECTION 2 Basic Switcher Control

### 2.3.2.12 IBMDSwitcher:: DoesSupportMultiViewVideoMode method

The **DoesSupportMultiViewVideoMode** method determines if a MultiView video standard is supported for a particular core video standard.

#### Syntax

```
HRESULT DoesSupportMultiViewVideoMode (BMDSwitcherVideoMode coreVideoMode, BMDSwitcherVideoMode  
multiviewVideoMode, boolean* supported);
```

#### Parameters

Name	Direction	Description
coreVideoMode	in	The core video standard.
multiviewVideoMode	in	The MultiView video standard to use for the coreVideoMode.
supported	out	Boolean value that is true if the MultiView video standard is supported for the coreVideoMode.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The supported parameter is invalid.
E_INVALIDARG	The multiviewVideoMode parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.2.13 IBMDSwitcher:: GetPowerStatus method

The **GetPowerStatus** method gets the connected power status, useful for models supporting multiple power sources.

#### Syntax

```
HRESULT GetPowerStatus (BMDSwitcherPowerStatus* powerStatus);
```

#### Parameters

Name	Direction	Description
powerStatus	out	The power status.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The powerStatus parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.14 IBMDSwitcher:: CreateIterator method

The **CreateIterator** method creates an iterator object interface for the specified interface ID.

#### Syntax

```
HRESULT CreateIterator (REFIID iid, LPVOID* ppv);
```

#### Parameters

Name	Direction	Description
iid	in	Iterator Interface ID to create an iterator for.
ppv	in	Pointer to return interface object.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The ppv parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to create interface object.
E_NOINTERFACE	Interface was not found.

## SECTION 2 Basic Switcher Control

### 2.3.2.15 IBMDSwitcher::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcher** object.

Pass an object implementing the **IBMDSwitcherCallback** interface to receive callbacks.

Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT      AddCallback (IBMDSwitcherCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.2.16 IBMDSwitcher::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.3

IBMDSwitcherCallback Interface

The **IBMDSwitcherCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcher** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An <b>IBMDSwitcherCallback</b> object interface is installed with <b>IBMDSwitcher::AddCallback</b> and removed with <b>IBMDSwitcher::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

## SECTION 2 Basic Switcher Control

### 2.3.3.1 **IBMDSwitcherCallback::Notify** method

The **Notify** is called when **IBMDSwitcher** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads.

Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

**HRESULT** Notify (BMDSwitcherEventType eventType, BMDSwitcherVideoMode coreVideoMode);

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherEventType</b> that describes the type of event that has occurred.
coreVideoMode	in	Video standard for which the event was triggered. This parameter is used in <b>bmdSwitcherEventTypeDownConverted</b> , <b>HDVideoModeChanged</b> and <b>bmdSwitcherEventTypeMultiViewVideoModeChanged</b> event types.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 2 Basic Switcher Control

### 2.3.4 IBMDSwitcherInputIterator Interface

The **IBMDSwitcherInputIterator** object interface is used to enumerate the available switcher inputs.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::CreateIterator</b> returns an <b>IBMDSwitcherInputIterator</b> object interface when the IID_IBMDSwitcherInputIterator IID is specified.

#### Public Member Functions

Method	Description
Next	Returns the next available switcher input.
GetById	Returns a switcher input for an input Id.

## SECTION 2 Basic Switcher Control

### 2.3.4.1 **IBMDSwitcherInputIterator::Next** method

The **Next** method returns the next available **IBMDSwitcherInput** object interface.

The **IBMDSwitcherInput** object interface must be released by the caller when no longer required.

#### Syntax

```
HRESULT Next (IBMDSwitcherInput** input);
```

#### Parameters

Name	Direction	Description
input	out	<b>IBMDSwitcherInput</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No (more) inputs are available.
E_POINTER	The input parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.4.2 IBMDSwitcherInputIterator::GetById method

The **GetById** method returns the **IBMDSwitcherInput** object interface corresponding to the specified Id.

#### Syntax

```
HRESULT GetById (BMDSwitcherInputId inputId, IBMDSwitcherInput** input);
```

#### Parameters

Name	Direction	Description
inputId	in	<b>BMDSwitcherInputId</b> of input.
input	out	<b>IBMDSwitcherInput</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The inputId parameter is invalid.
E_POINTER	The input parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.5

IBMDSwitcherInput Interface

The **IBMDSwitcherInput** object interface represents a physical switcher device.

**Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherInputDiscovery	IID_IBMDSwitcherInputDiscovery	An <b>IBMDSwitcherInput</b> object will be returned after a successful call to the <b>IBMDSwitcherInputDiscovery::ConnectTo</b> method

Public Member Functions	
Method	Description
AddCallback	Add a callback to receive input property changes.
RemoveCallback	Remove a callback.
GetInputId	Get the unique ID for this input.
GetPortType	Get the port type as a <b>BMDSwitcherPortType</b> .
GetInputAvailability	Get the outputs this input can be routed to, as a <b>BMDSwitcherInputAvailability</b> object.
SetShortName	Set the short name describing the switcher input as a string limited to 4 ASCII characters.
GetShortName	Get the short name describing the switcher input as a string limited to 4 ASCII characters.
SetLongName	Set the long name describing the switcher input as a Unicode string limited to 20 bytes.
GetLongName	Get the long name describing the switcher input as a Unicode string limited to 20 bytes.

## SECTION 2 Basic Switcher Control

Public Member Functions	
Method	Description
ResetNames	Reset the long and short names for this switcher input to the factory defaults for this input.
IsProgramTallied	Returns a flag indicating whether the input is currently program tallied.
IsPreviewTallied	Returns a flag indicating whether the input is currently preview tallied.
GetAvailableExternalPortTypes	Get the available external port types as a bit mask of <b>BMDSwitcherExternalPortType</b> .
SetCurrentExternalPortType	Set the external port type for this input using a <b>BMDSwitcherExternalPortType</b> .

## SECTION 2 Basic Switcher Control

### 2.3.5.1 IBMDSwitcherInput::AddCallback method

The **AddCallback** method configures a callback to be called when a switcher input property changes.

Adding a new callback will not affect previously added callbacks.

Callbacks will be called on a separate thread and in the order of their addition by **AddCallback**.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherInputCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.5.2 IBMDSwitcherInput::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherInputCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object to remove.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 2.3.5.3 IBMDSwitcherInput::GetInputId method

The **GetInputId** method gets the unique Id for the switcher input.

#### Syntax

```
HRESULT GetInputId (BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
inputId	out	unique Id for switcher input.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inputId parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.5.4 IBMDSwitcherInput::GetPortType method

The **GetPortType** method returns the port type of this switcher input as a **BMDSwitcherPortType**. This can be used to determine if this input is an external port (i.e. `bmdSwitcherPortTypeExternal`), or any of the internal port types such as color bars (i.e. `bmdSwitcherPortTypeColorBars`).

#### Syntax

```
HRESULT GetPortType(BMDSwitcherPortType* type);
```

#### Parameters

Name	Direction	Description
type	out	The port type.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The type parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.5 IBMDSwitcherInput::GetInputAvailability method

The **GetInputAvailability** method determines which outputs this input can be routed to. The available output groups are given as a bit mask of **BMDSwitcherInputAvailability**. The value returned can be bit-wise AND-ed with any **BMDSwitcherInputAvailability** value (e.g. `bmdSwitcherInputAvailabilityAuxOutputs`) to determine the availability of this input to that output group.

#### Syntax

```
HRESULT GetInputAvailability(BMDSwitcherInputAvailability* availability);
```

#### Parameters

Name	Direction	Description
availability	out	The availability of the input.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The availability parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.6 IBMDSwitcherInput::SetShortName method

The **SetShortName** method assigns the short name describing the switcher input as a string limited to 4 ASCII characters.

#### Syntax

```
HRESULT      SetShortName(string name);
```

#### Parameters

Name	Direction	Description
name	in	The short name for the switcher input, limited to 4 ASCII characters.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The name parameter is not a valid pointer.
E_INVALIDARG	The name parameter contains non-ASCII characters.



## SECTION 2 Basic Switcher Control

### 2.3.5.7 IBMDSwitcherInput::GetShortName method

The **GetShortName** method gets the short name describing the switcher input as a string limited to 4 ASCII characters.

#### Syntax

```
HRESULT GetShortName(string* name);
```

#### Parameters

Name	Direction	Description
name	out	The short name for the switcher input, limited to 4 ASCII characters.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The name parameter is not a valid pointer.

## SECTION 2 Basic Switcher Control

### 2.3.5.8 IBMDSwitcherInput::SetLongName method

The **SetLongName** method sets the long name, describing the switcher input as a Unicode string in UTF-8 format with a maximum length of 20 bytes. If a string longer than 20 bytes is provided, it will be truncated to the longest valid UTF-8 string of 20 bytes or less.

#### Syntax

```
HRESULT SetLongName(string name);
```

#### Parameters

Name	Direction	Description
name	in	The long name describing the switcher input as a Unicode string with a maximum length of 20 bytes.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The name parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.9 IBMDSwitcherInput::GetLongName method

The **GetLongName** method gets the long name for the switcher input, describing the input as a Unicode string in UTF-8 format with a maximum length of 20 bytes.

#### Syntax

```
HRESULT GetLongName(string* name);
```

#### Parameters

Name	Direction	Description
name	out	The long name describing the switcher input as a Unicode string with a maximum length of 20 bytes.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The name parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.10 IBMDSwitcherInput::ResetNames method

The **ResetNames** method resets the long and short names for this switcher input to the factory defaults for this input.

#### Syntax

```
HRESULT ResetNames();
```

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

## SECTION 2 Basic Switcher Control

### 2.3.5.11 IBMDSwitcherInput::IsProgramTallied method

The **IsProgramTallied** method determines whether this switcher input is currently program tallied.

#### Syntax

```
HRESULT IsProgramTallied(bool* isTallied);
```

#### Parameters

Name	Direction	Description
isTallied	out	Flag indicating if the input is currently program tallied.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The isTallied parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.12 IBMDSwitcherInput::IsPreviewTallied method

The **IsPreviewTallied** method determines whether this switcher input is currently preview tallied.

#### Syntax

```
HRESULT IsPreviewTallied(bool* isTallied);
```

#### Parameters

Name	Direction	Description
isTallied	out	Flag indicating if the input is currently preview tallied.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The isTallied parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.13 **IBMDSwitcherInput::GetAvailableExternalPortTypes** method

The **GetAvailableExternalPortTypes** method gets the available external port types for this switcher input, given as a bit mask of **BMDSwitcherExternalPortType**. This bit mask can be bit-wise AND-ed with any value of **BMDSwitcherExternalPortType** (e.g. `bmdSwitcherExternalPortTypeSDI`) to determine if that external port type is available for this input.

#### Syntax

```
HRESULT GetAvailableExternalPortTypes(BMDSwitcherExternalPortType* types);
```

#### Parameters

Name	Direction	Description
types	out	The available external port types for this switcher input as a bit mask of <b>BMDSwitcherExternalPortType</b> .

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The types parameter is not valid.

## SECTION 2 Basic Switcher Control

### 2.3.5.14 IBMDSwitcherInput::SetCurrentExternalPortType method

The **SetCurrentExternalPortType** method sets the external port type for this input using a **BMDSwitcherExternalPortType**. The external port type is settable only for some inputs and not all external port types are valid for a given input. Call the **GetAvailableExternalPortTypes** function to determine the available external port types for this input.

#### Syntax

```
HRESULT SetCurrentExternalPortType(BMDSwitcherExternalPortType type);
```

#### Parameters

Name	Direction	Description
type	in	The external port type.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	The type parameter is not valid a valid external port type for this input.



## SECTION 2 Basic Switcher Control

### 2.3.6 IBMDSwitcherInputCallback Interface

The **IBMDSwitcherInputCallback** object interface is a callback class which is called when a switcher input event such as a property change occurs.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInput	IID_IBMDSwitcherInput	An <b>IBMDSwitcherInputCallback</b> object interface may be installed with <b>IBMDSwitcherInput::AddCallback</b>

#### Public Member Functions

Method	Description
Notify	A Switcher Input event occurred such as a property change.

## SECTION 2 Basic Switcher Control

### 2.3.6.1 IBMDSwitcherInputCallback::Notify method

The **Notify** method is called when a **IBMDSwitcherInput** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads.

Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherInputEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherInputEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

SECTION

2

Basic Switcher Control

2.3.7

IBMDSwitcherMixEffectBlocklterator Interface

The **IBMDSwitcherMixEffectBlocklterator** object interface is used to enumerate the available mix effect blocks.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::Createlterator</b> returns an <b>IBMDSwitcherMixEffectBlocklterator</b> object interface when the IID_IBMDSwitcherMixEffectBlocklterator IID is specified.

Public Member Functions	
Method	Description
Next	Returns the next available switcher mix effect block.

## SECTION 2 Basic Switcher Control

### 2.3.7.1 **IBMDSwitcherMixEffectBlockIterator::Next** method

The **Next** method returns the next available **IBMDSwitcherMixEffectBlock** object interface.

The **IBMDSwitcherMixEffectBlock** object interface must be released by the caller when no longer required.

#### **Syntax**

```
HRESULT Next (IBMDSwitcherMixEffectBlock** mixEffectBlock);
```

#### **Parameters**

Name	Direction	Description
mixEffectBlock	out	<b>IBMDSwitcherMixEffectBlock</b> object interface

#### **Return Values**

Value	Description
S_OK	Success.
S_FALSE	No (more) mix effect blocks are available.
E_POINTER	The mixEffectBlock parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.8

IBMDSwitcherMixEffectBlock Interface

The **IBMDSwitcherMixEffectBlock** object interface represents a mix effect block of a switcher device.

**Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlockIterator	IID_IBMDSwitcherMixEffectBlockIterator	<b>IBMDSwitcherMixEffectBlockIterator::Next</b> returns <b>IBMDSwitcherMixEffectBlock</b> object interfaces for each available mix effect block of a switcher device.

Public Member Functions	
Method	Description
CreateIterator	Create an iterator.
AddCallback	Add a callback to receive mix effect block property changes.
RemoveCallback	Remove a callback.
GetFlag	Get the current value of a boolean encoding mode setting.
GetInt	Get the current value of a int64_t encoding mode setting.
GetFloat	Get the current value of a double encoding mode setting.
GetString	Get the current value of a string encoding mode setting.
SetFlag	Set the value for a boolean encoding mode setting.
SetInt	Set the value for an int64_t encoding mode setting.
SetFloat	Set the value for a double encoding mode setting.
SetString	Set the value for a string encoding mode setting.

## SECTION 2 Basic Switcher Control

### 2.3.8.1 IBMDSwitcherMixEffectBlock::CreateIterator method

The **CreateIterator** method creates an iterator object interface for the specified interface Id.

**Note:** In the current version of the Switcher SDK there are no supported iterator IIDs for this method.

#### Syntax

```
HRESULT CreateIterator (REFIID iid, LPVOID* ppv);
```

#### Parameters

Name	Direction	Description
iid	in	Iterator Interface ID to create an iterator for.
ppv	out	Pointer to return interface object.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The ppv parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to create interface object.
S_NOINTERFACE	Interface was not found.

## SECTION 2 Basic Switcher Control

### 2.3.8.2 IBMDSwitcherMixEffectBlock::AddCallback method

The **AddCallback** method configures a callback to be called when a mix effect block property changes.

Adding a new callback will not affect previously added callbacks. Callbacks will be called on a separate thread and in the order of their addition by **AddCallback**.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherMixEffectBlockCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMixEffectBlockCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.3 IBMDSwitcherMixEffectBlock::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherMixEffectBlockCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object to remove.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.8.4 **IBMDSwitcherMixEffectBlock::GetFlag** method

The **GetFlag** method gets the current value of the boolean property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT GetFlag (BMDSwitcherMixEffectBlockPropertyId propertyId, boolean* value);
```

#### Parameters

Name	Direction	Description
propertyId	in	<b>BMDSwitcherMixEffectBlockPropertyId</b> to get flag value.
value	out	The value corresponding to propertyId.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.5 **IBMDSwitcherMixEffectBlock::GetInt** method

The **GetInt** method gets the current value of the `int64_t` property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT          GetInt (BMDSwitcherMixEffectBlockPropertyId propertyId, int64_t* value);
```

#### Parameters

Name	Direction	Description
propertyId	in	<b>BMDSwitcherMixEffectBlockPropertyId</b> to get integer value.
value	out	The value corresponding to propertyId.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.6 **IBMDSwitcherMixEffectBlock::GetFloat** method

The **GetFloat** gets the current value of the double property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT GetFloat (BMDSwitcherMixEffectBlockPropertyId propertyId, double* value);
```

#### Parameters

Name	Direction	Description
propertyId	in	<b>BMDSwitcherMixEffectBlockPropertyId</b> to get double value.
value	out	The value corresponding to propertyId.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.7 IBMDSwitcherMixEffectBlock::GetString method

The **GetString** current value of the string property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

**HRESULT**            GetString (BMDSwitcherMixEffectBlockPropertyId propertyId, string value);

#### Parameters

Name	Direction	Description
propertyId	in	<b>BMDSwitcherMixEffectBlockPropertyId</b> to get string value.
value	out	The value corresponding to propertyId. This allocated string must be released by the caller when no longer required.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.
E_OUTOFMEMORY	Unable to allocate memory for string.

## SECTION 2 Basic Switcher Control

### 2.3.8.8 IBMDSwitcherMixEffectBlock::SetFlag method

The **SetFlag** method sets a boolean value into the property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT SetFlag (BMDSwitcherMixEffectBlockPropertyId propertyId, boolean value);
```

#### Parameters

Name	Direction	Description
propertyId	in	The ID of the property.
value	in	The boolean value to set into the selected property.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.9 IBMDSwitcherMixEffectBlock::SetInt method

The **SetInt** method sets an `int64_t` value into the property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT SetInt (BMDSwitcherMixEffectBlockPropertyId propertyId, int64_t value);
```

#### Parameters

Name	Direction	Description
propertyId	in	The ID of the property.
value	in	The integer value to set into the selected property.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.10 **IBMDSwitcherMixEffectBlock::SetFloat** method

The **SetFloat** method sets a double value into the property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT SetFloat (BMDSwitcherMixEffectBlockPropertyId propertyId, double value);
```

#### Parameters

Name	Direction	Description
propertyId	in	The ID of the property.
value	in	The double value to set into the selected property.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.

## SECTION 2 Basic Switcher Control

### 2.3.8.11 **IBMDSwitcherMixEffectBlock::SetString** method

The **SetString** method sets a string value into the property associated with the given **BMDSwitcherMixEffectBlockPropertyId**.

#### Syntax

```
HRESULT SetString (BMDSwitcherMixEffectBlockPropertyId propertyId, string value);
```

#### Parameters

Name	Direction	Description
propertyId	in	The ID of the property.
value	in	The string value to set into the selected property.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	One or more parameters are invalid.



## SECTION 2 Basic Switcher Control

### 2.3.8.12 **IBMDSwitcherMixEffectBlock::PerformAutoTransition** method

The **PerformAutoTransition** method initiates an automatic transition for the mix effect block.

When the transition begins the **bmdSwitcherMixEffectBlockPropertyIdInTransition** property will change to true and when the transition is complete it will become false. Throughout the transition the **bmdSwitcherMixEffectBlockPropertyIdTransitionPosition** and **bmdSwitcherMixEffectBlockPropertyIdTransitionFramesRemaining** properties will change to values corresponding to the progress through the transition.

#### **Syntax**

```
HRESULT PerformAutoTransition ();
```

#### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

## SECTION 2 Basic Switcher Control

### 2.3.8.13 **IBMDSwitcherMixEffectBlock::PerformCut** method

The **PerformCut** method initiates a cut for the mix effect block.

#### Syntax

```
HRESULT PerformCut ();
```

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 2.3.8.14 **IBMDSwitcherMixEffectBlock::PerformFadeToBlack** method

The **PerformFadeToBlack** method initiates a fade to black for the mix effect block.

When the fade to black begins the **bmdSwitcherMixEffectBlockPropertyIdInFadeToBlack** property will change to true and when the transition is complete it will become false. Throughout the fade to black the **bmdSwitcherMixEffectBlockPropertyIdFadeToBlackFramesRemaining** property will change to a value corresponding to the progress through the fade to black.

#### Syntax

```
HRESULT PerformFadeToBlack ();
```

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

## SECTION 2 Basic Switcher Control

### 2.3.9 IBMDSwitcherMixEffectBlockCallback Interface

The **IBMDSwitcherMixEffectBlockCallback** object interface is a callback class which is called when a mix effect block property changes.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherMixEffectBlockCallback</b> object interface may be installed with <b>IBMDSwitcherMixEffectBlock::AddCallback</b> .

#### Public Member Functions

Method	Description
PropertyChanged	Mix effect block property changed.

## SECTION 2 Basic Switcher Control

### 2.3.9.1 **IBMDSwitcherMixEffectBlockCallback::PropertyChanged** method

The **PropertyChanged** method is called when a mix effect block property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

**HRESULT**            **PropertyChanged** (**BMDSwitcherMixEffectBlockPropertyId** **propertyId**);

#### **Parameters**

Name	Direction	Description
propertyId	in	Id of the property that changed as a <b>BMDSwitcherMixEffectBlockPropertyId</b>

#### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

SECTION

2

Basic Switcher Control

2.3.10

IBMDSwitcherInputColor Interface

The **IBMDSwitcherInputColor** object interface is used for managing a color generator input port.

**Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherInput	IID_IBMDSwitcherInput	An <b>IBMDSwitcherInputColor</b> object interface can be obtained with <b>IBMDSwitcherInput::QueryInterface</b> .

Public Member Functions	
Method	Description
GetHue	Get the current hue value.
SetHue	Set the hue value.
GetSaturation	Get the current saturation value.
SetSaturation	Set the saturation value.
GetLuma	Get the current luminance value.
SetLuma	Set the luminance value.
AddCallback	Add a color input callback.
RemoveCallback	Remove a color input callback.

## SECTION 2 Basic Switcher Control

### 2.3.10.1 **IBMDSwitcherInputColor::GetHue** method

The **GetHue** method gets the current hue value.

#### **Syntax**

```
HRESULT      GetHue (double* hue);
```

#### **Parameters**

Name	Direction	Description
propertyId	in	Id of the property that changed as a <b>BMDSwitcherMixEffectBlockPropertyId</b>

#### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

## SECTION 2 Basic Switcher Control

### 2.3.10.2 IBMDSwitcherInputColor::SetHue method

The **SetHue** method sets the hue value.

#### Syntax

```
HRESULT SetHue (double hue);
```

#### Parameters

Name	Direction	Description
hue	in	The desired hue value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 2.3.10.3 IBMDSwitcherInputColor::GetSaturation method

The **GetSaturation** method gets the current hue value.

#### Syntax

```
HRESULT GetSaturation (double* sat);
```

#### Parameters

Name	Direction	Description
sat	out	The current sat value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The sat parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.10.4 IBMDSwitcherInputColor::SetSaturation method

The **SetSaturation** method sets the hue value.

#### Syntax

```
HRESULT SetSaturation (double sat);
```

#### Parameters

Name	Direction	Description
sat	in	The desired saturation value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 2.3.10.5 IBMDSwitcherInputColor::GetLuma method

The **GetLuma** method gets the current luminance value.

#### Syntax

```
HRESULT GetLuma (double* luma);
```

#### Parameters

Name	Direction	Description
luma	out	The current luminance value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The luma parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.10.6 IBMDSwitcherInputColor::SetLuma method

The **SetLuma** method sets the luminance value.

#### Syntax

```
HRESULT SetLuma (double luma);
```

#### Parameters

Name	Direction	Description
luma	in	The desired luminance value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 2 Basic Switcher Control

### 2.3.10.7 IBMDSwitcherInputColor::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherInputColor** object.

Pass an object implementing the **IBMDSwitcherInputColorCallback** interface to receive callbacks.

Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherInputColorCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputColorCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.10.8 IBMDSwitcherInputColor::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherInputColorCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputColorCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.11

IBMDSwitcherInputColorCallback Interface

The **IBMDSwitcherInputColorCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherInputColor** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputColor	IID_IBMDSwitcherInputColor	An <b>IBMDSwitcherInputColorCallback</b> object interface is installed with <b>IBMDSwitcherInputColor::AddCallback</b> and removed with <b>IBMDSwitcherInputColor::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

## SECTION 2 Basic Switcher Control

### 2.3.11.1 **IBMDSwitcherInputColorCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherInputColor** events occur, events such as a property change.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherInputColorEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherInputColorEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

SECTION

2

Basic Switcher Control

2.3.12

IBMDSwitcherInputAux Interface

The **IBMDSwitcherInputAux** object interface is used for managing an auxiliary output port.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInput	IID_IBMDSwitcherInput	An <b>IBMDSwitcherInputAux</b> object interface can be obtained with <b>IBMDSwitcherInput::QueryInterface</b> .

Public Member Functions	
Method	Description
GetInputSource	Get the selected input source.
SetInputSource	Select the input source.
GetInputAvailabilityMask	Get the corresponding <b>BMDSwitcherInputAvailability</b> bit value for auxiliary port.
AddCallback	Add an aux callback.
RemoveCallback	Remove an aux callback.

## SECTION 2 Basic Switcher Control

### 2.3.12.1 IBMDSwitcherInputAux::GetInputSource method

The **GetInputSource** method returns the currently selected input source.

#### Syntax

```
HRESULT GetInputSource (BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
inputId	out	The <b>BMDSwitcherInputId</b> of the selected input source.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid inputId parameter.

## SECTION 2 Basic Switcher Control

### 2.3.12.2 IBMDSwitcherInputAux::SetInputSource method

The **SetInputSource** method selects an input source for this auxiliary port.

#### Syntax

```
HRESULT SetInputSource (BMDSwitcherInputId inputId);
```

#### Parameters

Name	Direction	Description
inputId	in	The <b>BMDSwitcherInputId</b> of the desired input source.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	Invalid inputId parameter.
E_FAIL	Failure.



## SECTION 2 Basic Switcher Control

### 2.3.12.3 IBMDSwitcherInputAux::GetInputAvailabilityMask method

The GetInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for this auxiliary port. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value to determine whether an input is available for use as a source for this auxiliary port.

#### Syntax

```
HRESULT GetInputAvailabilityMask (BMDSwitcherInputAvailability* mask);
```

#### Parameters

Name	Direction	Description
mask	out	<b>BMDSwitcherInputAvailability</b> bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

## SECTION 2 Basic Switcher Control

### 2.3.12.4 IBMDSwitcherInputAux::AddCallback method

The AddCallback method configures a callback to be called when events occur for an **IBMDSwitcherInputAux** object. Pass an object implementing the **IBMDSwitcherInputAuxCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherInputAuxCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputAuxCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.12.5 IBMDSwitcherInputAux::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherInputAuxCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputAuxCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.13 IBMDSwitcherInputAuxCallback Interface

The **IBMDSwitcherInputAuxCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherInputAux** object. Like all callback methods, these callback methods may be called from another thread.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputAux	IID_IBMDSwitcherInputAux	An <b>IBMDSwitcherInputAuxCallback</b> object interface is installed with <b>IBMDSwitcherInputAux::AddCallback</b> and removed with <b>IBMDSwitcherInputAux::RemoveCallback</b>

#### Public Member Functions

Method	Description
Notify	Called when an event occurs.

## SECTION 2 Basic Switcher Control

### 2.3.13.1 **IBMDSwitcherInputAuxCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherInputAux** events occur, events such as a property change.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherInputAuxEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherInputAuxEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 2 Basic Switcher Control

### 2.3.14 **IBMDSwitcherMultiViewIterator** Interface

The **IBMDSwitcherMultiViewIterator** is used to enumerate the available MultiViews.

A reference to an **IBMDSwitcherMultiViewIterator** object interface may be obtained from an **IBMDSwitcher** object interface using the **CreateIterator** method. Pass **IID\_IBMDSwitcherMultiViewIterator** for the IID parameter.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::CreateIterator</b> can return an <b>IBMDSwitcherMultiViewIterator</b> object interface.

#### Public Member Functions

Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherMultiView</b> object interface.

## SECTION 2 Basic Switcher Control

### 2.3.14.1 **IBMDSwitcherMultiViewIterator::Next** method

The **Next** method returns the next available **IBMDSwitcherMultiView** object interface.

#### Syntax

```
HRESULT Next (IBMDSwitcherMultiView** multiView);
```

#### Parameters

Name	Direction	Description
multiView	out	<b>IBMDSwitcherMultiView</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherMultiView</b> objects available.
E_POINTER	The multiView parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.15

IBMDSwitcherMultiView Interface

The **IBMDSwitcherMultiView** object interface is used for accessing control functions of a MultiView output, such as setting up the layout format, or routing different inputs to windows.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMultiViewIterator	IID_IBMDSwitcherMultiViewIterator	An <b>IBMDSwitcherMultiView</b> object will be returned after a successful call to <b>IBMDSwitcherMultiViewIterator::Next</b> method.

Public Member Functions	
Method	Description
GetLayout	Get the current layout format.
SetLayout	Set layout format.
GetWindowInput	Get current input routing of a window.
SetWindowInput	Set input routing of a window.
GetWindowCount	Get number of windows available.
GetInputAvailabilityMask	Get the corresponding <b>BMDSwitcherInputAvailability</b> bit value for MultiView.
CanRouteInputs	Determine if this MultiView supports custom input-to-window routing.
AddCallback	Add a MultiView callback.
RemoveCallback	Remove a MultiView callback.



## SECTION 2 Basic Switcher Control

### 2.3.15.1 IBMDSwitcherMultiView::GetLayout method

The **GetLayout** method returns the current layout format.

#### Syntax

```
HRESULT GetLayout (BMDSwitcherMultiViewLayout* layout);
```

#### Parameters

Name	Direction	Description
layout	out	Current layout format as a <b>BMDSwitcherMultiViewLayout</b> .

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The layout parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.15.2 **BMDSwitcherMultiView::SetLayout** method

The **SetLayout** method sets the layout format.

#### Syntax

```
HRESULT SetLayout (BMDSwitcherMultiViewLayout* layout);
```

#### Parameters

Name	Direction	Description
layout	in	Desired layout format in <b>BMDSwitcherMultiViewLayout</b> .

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	Invalid layout parameter.

## SECTION 2 Basic Switcher Control

### 2.3.15.3 **IBMDSwitcherMultiView::GetWindowInput** method

The **GetWindowInput** method returns the current input source routed to the specified window.

#### Syntax

```
HRESULT GetWindowInput (uint32_t window, BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
window	in	Zero-based window index.
inputId	out	Input source as a <b>BMDSwitcherInputId</b>

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The window parameter is invalid.
E_POINTER	The inputId parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.15.4 IBMDSwitcherMultiView::SetWindowInput method

The **SetWindowInput** method routes an input source to the specified window.

#### Syntax

```
HRESULT SetWindowInput (uint32_t window, BMDSwitcherInputId inputId);
```

#### Parameters

Name	Direction	Description
window	in	Zero-based window index.
inputId	in	<b>BMDSwitcherInputId</b> input source.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The window and/or inputId parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.15.5 IBMDSwitcherMultiView::GetWindowCount method

The **GetWindowCount** method returns the total number of windows available to this MultiView.

#### Syntax

```
HRESULT GetWindowCount (uint32_t* windowCount);
```

#### Parameters

Name	Direction	Description
windowCount	out	Total number of windows.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The windowCount parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.15.6 IBMDSwitcherMultiView::GetInputAvailabilityMask method

The **GetInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for this MultiView. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value to determine whether an input is available for viewing in a window.

#### Syntax

```
HRESULT GetInputAvailabilityMask (BMDSwitcherInputAvailability* mask);
```

#### Parameters

Name	Direction	Description
mask	out	<b>BMDSwitcherInputAvailability</b> bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

## SECTION 2 Basic Switcher Control

### 2.3.15.7 IBMDSwitcherMultiView::CanRouteInputs method

The **CanRouteInputs** method returns whether this MultiView has custom input-to-window routing capability. This feature allows custom selection of input sources on each window, whereas without this feature the configuration is static and the window input sources cannot be changed. If the MultiView has no such capability, any call to **SetWindowInput** will fail.

#### Syntax

```
HRESULT CanRouteInputs (boolean* canRoute);
```

#### Parameters

Name	Direction	Description
canRoute	out	Boolean that indicates if this MultiView is capable of custom input-to-window routing.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The canRoute parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.15.8 **IBMDSwitcherMultiView::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherMultiView** object. Pass an object implementing the **IBMDSwitcherMultiViewCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherMultiViewCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMultiViewCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



## SECTION 2 Basic Switcher Control

### 2.3.15.9 IBMDSwitcherMultiView::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherMultiViewCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMultiViewCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.16 IBMDSwitcherMultiViewCallback Interface

The **IBMDSwitcherMultiViewCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherMultiView** object. Like all callback methods, these callback methods may be called from another thread.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMultiView	IID_IBMDSwitcherMultiView	An <b>IBMDSwitcherMultiViewCallback</b> object interface is installed with <b>IBMDSwitcherMultiView::AddCallback</b> and removed with <b>IBMDSwitcherMultiView::RemoveCallback</b>

#### Public Member Functions

Method	Description
Notify	Called when an event occurs.

## SECTION 2 Basic Switcher Control

### 2.3.16.1 **IBMDSwitcherMultiViewCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherMultiView** events occur, events such as a property change.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT** Notify (**BMDSwitcherMultiViewEventType** eventType, **int32\_t** window);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherMultiViewEventType</b> that describes the type of event that has occurred.
window	in	This parameter is only valid when eventType is <b>bmdSwitcherMultiViewEventTypeWindowChanged</b> , it specifies the window index that was changed.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 2 Basic Switcher Control

### 2.3.17 IBMDSwitcherSerialPortIterator Interface

The **IBMDSwitcherSerialPortIterator** object interface is used to enumerate the available serial ports on the switcher.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::CreateIterator</b> returns an <b>IBMDSwitcherSerialPortIterator</b> interface when the IID_IBMDSwitcherSerialPortIterator IID is specified.

#### Public Member Functions

Method	Description
Next	Returns the next available serial port.

## SECTION 2 Basic Switcher Control

### 2.3.17.1 IBMDSwitcherSerialPortIterator::Next method

The **Next** method returns the next available **IBMDSwitcherSerialPort** object interface.

The **IBMDSwitcherSerialPort** object interface must be released by the caller when no longer required.

#### Syntax

```
HRESULT      Next (IBMDSwitcherSerialPort** serialPort);
```

#### Parameters

Name	Direction	Description
serialPort	out	<b>IBMDSwitcherSerialPort</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No (more) serial ports available.
E_POINTER	The serialPort parameter is invalid.

# SECTION 2 Basic Switcher Control

## 2.3.18 IBMDSwitcherSerialPort Interface

The **IBMDSwitcherSerialPort** object interface is used for managing a serial port on the switcher.

### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSerialPortIterator	IID_ IBMDSwitcherSerialPortIterator	<b>IBMDSwitcherSerialPortIterator::Next</b> returns <b>IBMDSwitcherSerialPort</b> interfaces for each available serial port of a switcher device.

### Public Member Functions

Method	Description
SetFunction	Set the function of the serial port using a <b>BMDSwitcherSerialPortFunction</b> .
GetFunction	Returns the function of the serial port as a <b>BMDSwitcherSerialPortFunction</b> .
DoesSupportFunction	Check if a given <b>BMDSwitcherSerialPortFunction</b> is supported by the switcher.
AddCallback	Add a serial port callback.
RemoveCallback	Remove a serial port callback.

## SECTION 2 Basic Switcher Control

### 2.3.18.1 IBMDSwitcherSerialPort::SetFunction method

The **SetFunction** method sets the function of the serial port.

#### Syntax

```
HRESULT SetFunction (BMDSwitcherSerialPortFunction function);
```

#### Parameters

Name	Direction	Description
function	in	The function to which the serial port should be set.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The function parameter is not a valid serial port function.

## SECTION 2 Basic Switcher Control

### 2.3.18.2 **IBMDSwitcherSerialPort::GetFunction** method

The **GetFunction** method returns the current function of the serial port.

#### Syntax

```
HRESULT GetFunction (BMDSwitcherSerialPortFunction* function);
```

#### Parameters

Name	Direction	Description
function	out	A <b>BMDSwitcherSerialPortFunction</b> describing which function the serial port is currently set to.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The function parameter is not a valid pointer.



## SECTION 2 Basic Switcher Control

### 2.3.18.3 IBMDSwitcherSerialPort::DoesSupportFunction method

The **DoesSupportFunction** method is used to determine if a given serial port function is supported by the switcher.

#### Syntax

```
HRESULT DoesSupportFunction (BMDSwitcherSerialPortFunction function, boolean* supported);
```

#### Parameters

Name	Direction	Description
function	in	The serial port function being queried.
supported	out	Boolean value describing whether the specified function is supported by the switcher.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The supported parameter is not a valid pointer.
E_INVALIDARG	The function parameter is not a valid <b>BMDSwitcherSerialPortFunction</b> .

## SECTION 2 Basic Switcher Control

### 2.3.18.4 IBMDSwitcherSerialPort::AddCallback method

The **AddCallback** method configures a callback to be called when a switcher serial port property changes, such as a change in the serial port function.

Adding a new callback will not affect previously added callbacks.

Callbacks will be called on a separate thread and in the order of their addition by **AddCallback**.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherSerialPortCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherSerialPortCallback</b> interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## SECTION 2 Basic Switcher Control

### 2.3.18.5 IBMDSwitcherSerialPort::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherSerialPortCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object to remove.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

SECTION

2

Basic Switcher Control

2.3.19

IBMDSwitcherSerialPortCallback Interface

The **IBMDSwitcherSerialPortCallback** object interface is a callback class which is called when a switcher serial port event occurs, such as a change in the serial port function. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSerialPort	IID_IBMDSwitcherSerialPort	An <b>IBMDSwitcherSerialPortCallback</b> interface may be installed with <b>IBMDSwitcherSerialPort::AddCalback</b>

Public Member Functions	
Method	Description
Notify	A Switcher Serial Port event occurred, such as a change in the serial port function.

## SECTION 2 Basic Switcher Control

### 2.3.19.1 **IBMDSwitcherSerialPortCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherSerialPort** events occur, such as a change in the serial port function.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads.

Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherSerialPortEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherSerialPortEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_FAIL	Failure.
S_OK	Success.

# SECTION 3 Advanced Transitions

## 3 Advanced Transitions

Transitions can be more advanced than making a simple cut between sources. This API provides a plethora of methods to enhance how a transition is performed.

### 3.1 Data Types

#### 3.1.1 Mix Parameters Event Type

**BMDSwitcherTransitionMixParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionMixParameters**.

<b>bmdSwitcherTransitionMixParametersEventTypeRateChanged</b>	The rate changed.
---	-------------------

#### 3.1.2 Dip Parameters Event Type

**BMDSwitcherTransitionDipParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionDipParameters**.

<b>bmdSwitcherTransitionDipParametersEventTypeRateChanged</b>	The rate changed.
---	-------------------

<b>bmdSwitcherTransitionDipParametersEventTypeInputDipChanged</b>	The dip input changed.
---	------------------------

#### 3.1.3 Wipe Parameters Event Type

**BMDSwitcherTransitionWipeParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionWipeParameters**.

<b>bmdSwitcherTransitionWipeParametersEventTypeRateChanged</b>	The rate changed.
--	-------------------

<b>bmdSwitcherTransitionWipeParametersEventTypePatternChanged</b>	The pattern changed.
---	----------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeBorderSizeChanged</b>	The border size changed.
--	--------------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeInputBorderChanged</b>	The border input changed.
---	---------------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeSymmetryChanged</b>	The symmetry changed.
--	-----------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeSoftnessChanged</b>	The softness changed.
--	-----------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeHorizontalOffsetChanged</b>	The horizontal offset changed.
--	--------------------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeVerticalOffsetChanged</b>	The vertical offset changed.
--	------------------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeReverseChanged</b>	The reverse flag changed.
---	---------------------------

<b>bmdSwitcherTransitionWipeParametersEventTypeFlipFlopChanged</b>	The flip flop flag changed.
--	-----------------------------

# SECTION 3

## Advanced Transitions

### 3.1.4 DVE Parameters Event Type

**BMDSwitcherTransitionDVEParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionDVEParameters**.

<b>bmdSwitcherTransitionDVEParametersEventTypeRateChanged</b>	The rate changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeLogoRateChanged</b>	The logo rate changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeReverseChanged</b>	The reverse flag changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeFlipFlopChanged</b>	The flip flop flag changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeStyleChanged</b>	The style changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeInputFillChanged</b>	The fill input changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeInputCutChanged</b>	The cut input changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeEnableKeyChanged</b>	The enable key flag changed.
<b>bmdSwitcherTransitionDVEParametersEventTypePreMultipliedChanged</b>	The pre-multiplied flag changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeClipChanged</b>	The clip changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeGainChanged</b>	The gain changed.
<b>bmdSwitcherTransitionDVEParametersEventTypeInverseChanged</b>	The inverse flag changed.

# SECTION 3

## Advanced Transitions

### 3.1.5 Stinger Parameters Event Type

**BMDSwitcherTransitionStingerParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionStingerParameters**.

<b>bmdSwitcherTransitionStingerParametersEventTypeSourceChanged</b>	The source changed.
<b>bmdSwitcherTransitionStingerParametersEventTypePreMultipliedChanged</b>	The pre-multiplied flag changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeClipChanged</b>	The clip changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeGainChanged</b>	The gain changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeInverseChanged</b>	The inverse flag changed.
<b>bmdSwitcherTransitionStingerParametersEventTypePrerollChanged</b>	The preroll changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeClipDurationChanged</b>	The clip duration changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeTriggerPointChanged</b>	The trigger point changed.
<b>bmdSwitcherTransitionStingerParametersEventTypeMixRateChanged</b>	The mix rate changed.

### 3.1.6 Transition Parameters Event Type

**BMDSwitcherTransitionParametersEventType** enumerates the possible event types for **IBMDSwitcherTransitionParameters**.

<b>bmdSwitcherTransitionParametersEventTypeTransitionStyleChanged</b>	The transition style changed.
<b>bmdSwitcherTransitionParametersEventTypeNextTransitionStyleChanged</b>	The next transition style changed.
<b>bmdSwitcherTransitionParametersEventTypeTransitionSelectionChanged</b>	The transition selection changed.
<b>bmdSwitcherTransitionParametersEventTypeNextTransitionSelectionChanged</b>	The next transition selection changed.



# SECTION 3 Advanced Transitions

## 3.1.7 Transition Style

**BMDSwitcherTransitionStyle** enumerates the possible transition styles, used by the **IBMDSwitcherTransitionParameters** object interface.

<b>bmdSwitcherTransitionStyleMix</b>	Mix style.
<b>bmdSwitcherTransitionStyleDip</b>	Dip style.
<b>bmdSwitcherTransitionStyleWipe</b>	Wipe style.
<b>bmdSwitcherTransitionStyleDVE</b>	DVE style.
<b>bmdSwitcherTransitionStyleStinger</b>	Stinger style.

## 3.1.8 Transition Selection

**BMDSwitcherTransitionSelection** is a bit set that enumerates what to include in a transition. This type is used by **IBMDSwitcherTransitionParameters** and **IBMDSwitcherKey** object interfaces.

<b>bmdSwitcherTransitionSelectionBackground</b>	Include background in transition.
<b>bmdSwitcherTransitionSelectionKey1</b>	Include key 1 in transition.
<b>bmdSwitcherTransitionSelectionKey2</b>	Include key 2 in transition.
<b>bmdSwitcherTransitionSelectionKey3</b>	Include key 3 in transition.
<b>bmdSwitcherTransitionSelectionKey4</b>	Include key 4 in transition.

# SECTION 3 Advanced Transitions

## 3.1.9 DVE Transition Style

**BMDSwitcherDVETransitionStyle** enumerates the possible transition styles.

This type is used by the **IBMDSwitcherTransitionDVEParameters** object interface.

<b>bmdSwitcherDVETransitionStyleSwooshTopLeft</b>	Top left swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshTop</b>	Top swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshTopRight</b>	Top right swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshLeft</b>	Left swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshRight</b>	Right swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshBottomLeft</b>	Bottom left swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshBottom</b>	Bottom swoosh.
<b>bmdSwitcherDVETransitionStyleSwooshBottomRight</b>	Bottom right swoosh.
<b>bmdSwitcherDVETransitionStyleSpinCWTopLeft</b>	Top left clockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCWTopRight</b>	Top right clockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCWBottomLeft</b>	Bottom left clockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCWBottomRight</b>	Bottom right clockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCCWTopLeft</b>	Top left counterclockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCCWTopRight</b>	Top right counterclockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCCWBottomLeft</b>	Bottom left counterclockwise spin.
<b>bmdSwitcherDVETransitionStyleSpinCCWBottomRight</b>	Bottom right counterclockwise spin.
<b>bmdSwitcherDVETransitionStyleSqueezeTopLeft</b>	Top left squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeTop</b>	Top squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeTopRight</b>	Top right squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeLeft</b>	Left squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeRight</b>	Right squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeBottomLeft</b>	Bottom left squeeze.

## SECTION 3 Advanced Transitions

<b>bmdSwitcherDVETransitionStyleSqueezeBottom</b>	Bottom squeeze.
<b>bmdSwitcherDVETransitionStyleSqueezeBottomRight</b>	Bottom right squeeze.
<b>bmdSwitcherDVETransitionStylePushTopLeft</b>	Top left push.
<b>bmdSwitcherDVETransitionStylePushTop</b>	Top push.
<b>bmdSwitcherDVETransitionStylePushTopRight</b>	Top right push.
<b>bmdSwitcherDVETransitionStylePushLeft</b>	Left push.
<b>bmdSwitcherDVETransitionStylePushRight</b>	Right push.
<b>bmdSwitcherDVETransitionStylePushBottomLeft</b>	Bottom left push.
<b>bmdSwitcherDVETransitionStylePushBottom</b>	Bottom push.
<b>bmdSwitcherDVETransitionStylePushBottomRight</b>	Bottom right push.
<b>bmdSwitcherDVETransitionStyleGraphicCWSpin</b>	Clockwise graphic spin.
<b>bmdSwitcherDVETransitionStyleGraphicCCWSpin</b>	Counterclockwise graphic spin.
<b>bmdSwitcherDVETransitionStyleGraphicLogoWipe</b>	Graphic logo wipe.

### 3.1.10 Stinger Transition Source

**BMDSwitcherStingerTransitionSource** enumerates the possible transition sources. This type is used by the **IBMDSwitcherTransitionStingerParameters** object interface.

<b>bmdSwitcherStingerTransitionSourceMediaPlayer1</b>	Media player 1.
<b>bmdSwitcherStingerTransitionSourceMediaPlayer2</b>	Media player 2.
<b>bmdSwitcherStingerTransitionSourceMediaPlayer3</b>	Media player 3.
<b>bmdSwitcherStingerTransitionSourceMediaPlayer4</b>	Media player 4.
<b>bmdSwitcherStingerTransitionSourceNone</b>	None.

3.2

Interface Reference

3.2.1

IBMDSwitcherTransitionMixParameters Interface

The IBMDSwitcherTransitionMixParameters object interface is used for manipulating transition settings specific to mix parameters.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherTransitionMixParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

Public Member Functions	
Method	Description
GetRate	Get the current rate.
SetRate	Set the rate.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

# SECTION 3 Advanced Transitions

## 3.2.1.1 IBMDSwitcherTransitionMixParameters::GetRate method

The **GetRate** method returns the current rate in frames.

### Syntax

```
HRESULT GetRate (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	out	The current rate.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.1.2 IBMDSwitcherTransitionMixParameters::SetRate method

The **SetRate** method sets the rate in frames.

### Syntax

```
HRESULT SetRate (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The frames parameter is invalid.

### 3.2.1.3 IBMDSwitcherTransitionMixParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionMixParameters** object. Pass an object implementing the **IBMDSwitcherTransitionMixParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT      AddCallback (IBMDSwitcherTransitionMixParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionMixParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.1.4 IBMDSwitcherTransitionMixParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTransitionMixParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionMixParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



3.2.2

IBMDSwitcherTransitionMixParametersCallback Interface

The **IBMDSwitcherTransitionMixParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionMixParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionMixParameters	IID_ IBMDSwitcherTransitionMixParameters	An <b>IBMDSwitcherTransitionMixParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionMixParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionMixParameters::RemoveCallback</b>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

### 3.2.2.1 IBMDSwitcherTransitionMixParametersCallback::Notify method

The **Notify** method is called when **IBMDSwitcherTransitionMixParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherTransitionMixParametersEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionMixParametersEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3

## Advanced Transitions

### 3.2.3 IBMDSwitcherTransitionDipParameters Interface

The **IBMDSwitcherTransitionDipParameters** object interface is used for manipulating transition settings specific to dip parameters.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherTransitionDipParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

#### Public Member Functions

Method	Description
GetRate	Get the current rate.
SetRate	Set the rate.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

# SECTION 3

## Advanced Transitions

### 3.2.3.1 IBMDSwitcherTransitionDipParameters::GetRate method

The **GetRate** method returns the current rate in frames.

#### Syntax

```
HRESULT GetRate (uint32_t* frames);
```

#### Parameters

Name	Direction	Description
frames	out	The current rate.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.3.2 IBMDSwitcherTransitionDipParameters::SetRate method

The **SetRate** method sets the rate in frames.

### Syntax

```
HRESULT SetRate (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The frames parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.3.3 IBMDSwitcherTransitionDipParameters::GetInputDip method

The **GetInputDip** method returns the current dip input.

### Syntax

```
HRESULT GetInputDip (BMDSwitcherInputId* input);
```

### Parameters

Name	Direction	Description
input	out	The current dip input.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.3.4 IBMDSwitcherTransitionDipParameters::SetInputDip method

The **SetInputDip** method sets the dip input.

### Syntax

```
HRESULT SetInputDip (BMDSwitcherInputId input);
```

### Parameters

Name	Direction	Description
input	in	The desired dip input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

## SECTION 3 Advanced Transitions

### 3.2.3.5 IBMDSwitcherTransitionDipParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionDipParameters** object. Pass an object implementing the **IBMDSwitcherTransitionDipParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherTransitionDipParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionDipParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



# SECTION 3

## Advanced Transitions

### 3.2.3.6 IBMDSwitcherTransitionDipParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTransitionDipParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionDipParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

3.2.4

IBMDSwitcherTransitionDipParametersCallback Interface

The **IBMDSwitcherTransitionDipParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionDipParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionDipParameters	IID_IBMDSwitcherTransitionDipParameters	An <b>IBMDSwitcherTransitionDipParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionDipParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionDipParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

# SECTION 3 Advanced Transitions

## 3.2.4.1 IBMDSwitcherTransitionDipParametersCallback::Notify method

The **Notify** method is called when **IBMDSwitcherTransitionDipParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

### Syntax

```
HRESULT Notify (BMDSwitcherTransitionDipParametersEventType eventType);
```

### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionDipParametersEventType</b> that describes the type of event that has occurred.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

3.2.5

IBMDSwitcherTransitionWipeParametersCallback Interface

The **IBMDSwitcherTransitionWipeParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionWipeParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionWipeParameters	IID_IBMDSwitcherTransitionWipeParameters	<b>IBMDSwitcherTransitionWipeParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionWipeParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionWipeParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

# SECTION 3 Advanced Transitions

## 3.2.5.1 IBMDSwitcherTransitionWipeParametersCallback::Notify method

The **Notify** method is called when **IBMDSwitcherTransitionWipeParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

### Syntax

```
HRESULT Notify (BMDSwitcherTransitionWipeParametersEventType eventType);
```

### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionWipeParametersEventType</b> that describes the type of event that has occurred.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3

## Advanced Transitions

### 3.2.6 IBMDSwitcherTransitionWipeParameters Interface

The **IBMDSwitcherTransitionWipeParameters** object interface is used for manipulating transition settings specific to wipe parameters.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherTransitionWipeParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

#### Public Member Functions

Method	Description
GetRate	Get the current rate.
SetRate	Set the rate.
GetPattern	Get the current pattern.
SetPattern	Set the pattern.
GetBorderSize	Get the current border size.
SetBorderSize	Set the border size.
GetInputBorder	Get the current border input.
SetInputBorder	Set the border input.
GetSymmetry	Get the current symmetry.
SetSymmetry	Set the symmetry.
GetSoftness	Get the current softness.
SetSoftness	Set the softness.
GetHorizontalOffset	Get the current horizontal offset.
SetHorizontalOffset	Set the horizontal offset.

# SECTION 3

## Advanced Transitions

Public Member Functions	
Method	Description
GetVerticalOffset	Get the current vertical offset.
SetVerticalOffset	Set the vertical offset.
GetReverse	Get the current reverse flag.
SetReverse	Set the reverse flag.
GetFlipFlop	Get the current flip flop flag.
SetFlipFlop	Set the flip flop flag.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

# SECTION 3 Advanced Transitions

## 3.2.6.1 IBMDSwitcherTransitionWipeParameters::GetRate method

The **GetRate** method returns the current rate in frames.

### Syntax

```
HRESULT GetRate (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	out	The current rate.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.



# SECTION 3 Advanced Transitions

## 3.2.6.2 IBMDSwitcherTransitionWipeParameters::SetRate method

The **SetRate** method sets the rate in frames.

### Syntax

```
HRESULT SetRate (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The frames parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.6.3 IBMDSwitcherTransitionWipeParameters::GetPattern method

The **GetPattern** method returns the current pattern style.

### Syntax

```
HRESULT GetPattern (BMDSwitcherPatternStyle* pattern);
```

### Parameters

Name	Direction	Description
pattern	out	The current pattern.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The pattern parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.6.4 IBMDSwitcherTransitionWipeParameters::SetPattern method

The **SetPattern** method sets the rate in frames.

### Syntax

```
HRESULT SetPattern (BMDSwitcherPatternStyle pattern);
```

### Parameters

Name	Direction	Description
frames	in	The desired pattern.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The pattern parameter is invalid.

## SECTION 3 Advanced Transitions

### 3.2.6.5 IBMDSwitcherTransitionWipeParameters::GetBorderSize method

The **GetBorderSize** method returns the current border size.

#### Syntax

```
HRESULT GetBorderSize (double* size);
```

#### Parameters

Name	Direction	Description
size	out	The current border size.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The size parameter is invalid.

### 3.2.6.6 IBMDSwitcherTransitionWipeParameters::SetBorderSize method

The **SetBorderSize** method sets the border size.

#### Syntax

```
HRESULT SetBorderSize (double size);
```

#### Parameters

Name	Direction	Description
size	in	The desired border size.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3

## Advanced Transitions

### 3.2.6.7 IBMDSwitcherTransitionWipeParameters::GetInputBorder method

The **GetInputBorder** method returns the current border input.

#### Syntax

```
HRESULT GetInputBorder (BMDSwitcherInputId* input);
```

#### Parameters

Name	Direction	Description
input	out	The current border input.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.6.8 IBMDSwitcherTransitionWipeParameters::SetInputBorder method

The **SetInputBorder** method sets the border input.

### Syntax

```
HRESULT SetInputBorder (BMDSwitcherInputId input);
```

### Parameters

Name	Direction	Description
input	in	The desired border input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

## SECTION 3 Advanced Transitions

### 3.2.6.9 IBMDSwitcherTransitionWipeParameters::GetSymmetry method

The **GetSymmetry** method returns the current symmetry.

#### Syntax

```
HRESULT GetSymmetry (double* symmetry);
```

#### Parameters

Name	Direction	Description
symmetry	out	The current symmetry.

#### Return Values

Value	Description
S_OK	Success.

### 3.2.6.10 IBMDSwitcherTransitionWipeParameters::SetSymmetry method

The **SetSymmetry** method sets the symmetry.

#### Syntax

```
HRESULT SetSymmetry (double symmetry);
```

#### Parameters

Name	Direction	Description
symmetry	in	The desired symmetry.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.6.11 IBMDSwitcherTransitionWipeParameters::GetSoftness method

The **GetSoftness** method returns the current softness.

#### Syntax

```
HRESULT GetSoftness (double* soft);
```

#### Parameters

Name	Direction	Description
soft	out	The current softness.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The soft parameter is invalid.

### 3.2.6.12 IBMDSwitcherTransitionWipeParameters::SetSoftness method

The **SetSoftness** method sets the softness.

#### Syntax

```
HRESULT SetSoftness (double soft);
```

#### Parameters

Name	Direction	Description
soft	in	The desired softness.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



# SECTION 3 Advanced Transitions

## 3.2.6.13 IBMDSwitcherTransitionWipeParameters::GetHorizontalOffset method

The **GetHorizontalOffset** method returns the current horizontal offset.

### Syntax

```
HRESULT GetHorizontalOffset (double* hOffset);
```

### Parameters

Name	Direction	Description
hOffset	out	The current horizontal offset.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The hOffset parameter is invalid.

## 3.2.6.14 IBMDSwitcherTransitionWipeParameters::SetHorizontalOffset method

The **SetHorizontalOffset** method sets the horizontal offset.

### Syntax

```
HRESULT SetHorizontalOffset (double hOffset);
```

### Parameters

Name	Direction	Description
hOffset	in	The desired horizontal offset.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

3.2.6.15 **IBMDSwitcherTransitionWipeParameters::GetVerticalOffset** method

The **GetVerticalOffset** method returns the current vertical offset.

**Syntax**

```
HRESULT      GetVerticalOffset (double* vOffset);
```

**Parameters**

Name	Direction	Description
vOffset	out	The current vertical offset.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The vOffset parameter is invalid.

3.2.6.16 **IBMDSwitcherTransitionWipeParameters::SetVerticalOffset** method

The **SetVerticalOffset** method sets the vertical offset.

**Syntax**

```
HRESULT      SetVerticalOffset (double vOffset);
```

**Parameters**

Name	Direction	Description
vOffset	in	The desired vertical offset.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.6.17 IBMDSwitcherTransitionWipeParameters::GetReverse method

The **GetReverse** method returns the current reverse flag.

#### Syntax

```
HRESULT GetReverse (boolean* reverse);
```

#### Parameters

Name	Direction	Description
reverse	out	The current reverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The reverse parameter is invalid.

### 3.2.6.18 IBMDSwitcherTransitionWipeParameters::SetReverse method

The **SetReverse** method sets the reverse flag.

#### Syntax

```
HRESULT SetReverse (boolean reverse);
```

#### Parameters

Name	Direction	Description
reverse	in	The desired reverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

3.2.6.19 **IBMDSwitcherTransitionWipeParameters::GetFlipFlop method**

The **GetFlipFlop** method returns the current flip flop flag.

**Syntax**

```
HRESULT          GetFlipFlop (boolean* flipflop);
```

**Parameters**

Name	Direction	Description
flipflop	out	The current flip flop flag.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The flipflop parameter is invalid.

3.2.6.20 **IBMDSwitcherTransitionWipeParameters::SetFlipFlop method**

The **SetFlipFlop** method sets the flip flop flag.

**Syntax**

```
HRESULT          SetFlipFlop (boolean flipflop);
```

**Parameters**

Name	Direction	Description
flipflop	in	The desired flip flop flag.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 3.2.6.21 IBMDSwitcherTransitionWipeParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionWipeParameters** object. Pass an object implementing the **IBMDSwitcherTransitionWipeParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT      AddCallback (IBMDSwitcherTransitionWipeParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionWipeParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.6.22 IBMDSwitcherTransitionWipeParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTransitionWipeParametersCallback* allback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionWipeParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7 IBMDSwitcherTransitionDVEParameters Interface

The **IBMDSwitcherTransitionDVEParameters** object interface is used for manipulating transition settings specific to DVE parameters.

### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherTransitionDVEParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

Public Member Functions	
Method	Description
GetRate	Get the current rate.
SetRate	Set the rate.
GetLogoRate	Get the current logo rate.
SetLogoRate	Set the logo rate.
GetReverse	Get the current reverse flag.
SetReverse	Set the reverse flag.
GetFlipFlop	Get the current flip flop flag.
SetFlipFlop	Set the flip flop flag.
GetStyle	Get the current style.
SetStyle	Set the style.
GetInputFill	Get the current fill input.
SetInputFill	Set the fill input.
GetInputCut	Get the current cut input.
SetInputCut	Set the cut input.

# SECTION 3

## Advanced Transitions

Public Member Functions	
Method	Description
GetFillInputAvailabilityMask	Get the availability mask for the fill of this input.
GetCutInputAvailabilityMask	Get the availability mask for the cut of this input.
GetEnableKey	Get the current enable key.
SetEnableKey	Set the enable key.
GetPreMultiplied	Get the current pre-multiplied flag.
SetPreMultiplied	Set the pre-multiplied flag.
GetClip	Get the current clip value.
SetClip	Set the clip value.
GetGain	Get the current gain.
SetGain	Set the gain.
GetInverse	Get the current inverse flag.
SetInverse	Set the inverse flag.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.



# SECTION 3 Advanced Transitions

## 3.2.7.1 IBMDSwitcherTransitionDVEParameters::GetRate method

The **GetRate** method returns the current rate in frames.

### Syntax

```
HRESULT GetRate (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	out	The current rate.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7.2 IBMDSwitcherTransitionDVEParameters::SetRate method

The **SetRate** method sets the rate in frames.

### Syntax

```
HRESULT SetRate (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The frames parameter is invalid.

## SECTION 3 Advanced Transitions

### 3.2.7.3 IBMDSwitcherTransitionDVEParameters::GetLogoRate method

The **GetLogoRate** method returns the current logo rate in frames.

#### Syntax

```
HRESULT GetLogoRate (uint32_t* frames);
```

#### Parameters

Name	Direction	Description
frames	out	The current logo rate.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.7.4 IBMDSwitcherTransitionDVEParameters::SetLogoRate method

The **SetLogoRate** method sets the logo rate in frames.

#### Syntax

```
HRESULT SetLogoRate (uint32_t frames);
```

#### Parameters

Name	Direction	Description
frames	in	The desired logo rate in frames.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The frames parameter is invalid.

3.2.7.5 **IBMDSwitcherTransitionDVEParameters::GetReverse** method

The **GetReverse** method returns the current reverse flag.

**Syntax**

```
HRESULT      GetReverse (boolean* reverse);
```

**Parameters**

Name	Direction	Description
reverse	out	The current reverse flag.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The reverse parameter is invalid.

3.2.7.6 **IBMDSwitcherTransitionDVEParameters::SetReverse** method

The **SetReverse** method sets the reverse flag.

**Syntax**

```
HRESULT      SetReverse (boolean reverse);
```

**Parameters**

Name	Direction	Description
reverse	in	The desired reverse flag.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.7.7 IBMDSwitcherTransitionDVEParameters::GetFlipFlop method

The **GetFlipFlop** method returns the current flip flop flag.

#### Syntax

```
HRESULT GetFlipFlop (boolean* flipflop);
```

#### Parameters

Name	Direction	Description
flipflop	out	The current flip flop flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The flipflop parameter is invalid.

### 3.2.7.8 IBMDSwitcherTransitionDVEParameters::SetFlipFlop method

The **SetFlipFlop** method sets the flip flop flag.

#### Syntax

```
HRESULT SetFlipFlop (boolean flipflop);
```

#### Parameters

Name	Direction	Description
flipflop	in	The desired flip flop flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.7.9 IBMDSwitcherTransitionDVEParameters::GetStyle method

The **GetStyle** method returns the current style.

### Syntax

```
HRESULT GetStyle (BMDSwitcherDVETransitionStyle* style);
```

### Parameters

Name	Direction	Description
style	out	The current style.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The style parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7.10 IBMDSwitcherTransitionDVEParameters::SetStyle method

The **SetStyle** method sets the style.

### Syntax

```
HRESULT SetStyle (BMDSwitcherDVETransitionStyle style);
```

### Parameters

Name	Direction	Description
style	in	The desired style.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The style parameter is invalid.



# SECTION 3 Advanced Transitions

## 3.2.7.11 IBMDSwitcherTransitionDVEParameters::GetInputFill method

The **GetInputFill** method returns the current fill input.

### Syntax

```
HRESULT GetInputFill (BMDSwitcherInputId* input);
```

### Parameters

Name	Direction	Description
input	out	The current fill input.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7.12 IBMDSwitcherTransitionDVEParameters::SetInputFill method

The **SetInputFill** method sets the fill input.

### Syntax

```
HRESULT SetInputFill (BMDSwitcherInputId input);
```

### Parameters

Name	Direction	Description
input	in	The desired fill input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7.13 IBMDSwitcherTransitionDVEParameters::GetInputCut method

The **GetInputCut** method returns the current cut input.

### Syntax

```
HRESULT GetInputCut (BMDSwitcherInputId* input);
```

### Parameters

Name	Direction	Description
input	out	The current cut input.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.7.14 IBMDSwitcherTransitionDVEParameters::SetInputCut method

The **SetInputCut** method sets the cut input.

### Syntax

```
HRESULT SetInputCut (BMDSwitcherInputId input);
```

### Parameters

Name	Direction	Description
input	in	The desired cut input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

### 3.2.7.15 IBMDSwitcherTransitionDVEParameters::GetFillInputAvailabilityMask method

The **GetFillInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for fill inputs available to this DVE transition. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a fill input for this DVE transition.

#### Syntax

```
HRESULT GetFillInputAvailabilityMask (BMDSwitcherInputAvailability* mask);
```

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

3.2.7.16

IBMDSwitcherTransitionDVEParameters::GetCutInputAvailabilityMask method

The **GetCutInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for cut inputs available to this DVE transition. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a cut input for this DVE transition.

**Syntax**

**HRESULT**            GetCutInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

**Parameters**

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

### 3.2.7.17 IBMDSwitcherTransitionDVEParameters::GetEnableKey method

The **GetEnableKey** method returns the current enableKey flag.

#### Syntax

```
HRESULT      GetEnableKey (boolean* enableKey);
```

#### Parameters

Name	Direction	Description
enableKey	out	The current enableKey flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The enableKey parameter is invalid.

### 3.2.7.18 IBMDSwitcherTransitionDVEParameters::SetEnableKey method

The **SetEnableKey** method sets the enableKey flag.

#### Syntax

```
HRESULT      SetEnableKey (boolean enableKey);
```

#### Parameters

Name	Direction	Description
enableKey	in	The desired enableKey flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

3.2.7.19 **IBMDSwitcherTransitionDVEParameters::GetPreMultiplied method**

The **GetPreMultiplied** method returns the current pre-multiplied flag.

**Syntax**

```
HRESULT      GetPreMultiplied (boolean* preMultiplied);
```

**Parameters**

Name	Direction	Description
preMultiplied	out	The current pre-multiplied flag.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The preMultiplied parameter is invalid.

3.2.7.20 **IBMDSwitcherTransitionDVEParameters::SetPreMultiplied method**

The **GetPreMultiplied** method sets the pre-multiplied flag.

**Syntax**

```
HRESULT      SetPreMultiplied (boolean preMultiplied);
```

**Parameters**

Name	Direction	Description
preMultiplied	in	The desired pre-multiplied flag.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



## SECTION 3 Advanced Transitions

### 3.2.7.21 IBMDSwitcherTransitionDVEParameters::GetClip method

The **GetClip** method returns the current clip value.

#### Syntax

```
HRESULT GetClip (double* clip);
```

#### Parameters

Name	Direction	Description
clip	out	The current clip value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The clip parameter is invalid.

### 3.2.7.22 IBMDSwitcherTransitionDVEParameters::SetClip method

The **SetClip** method sets the clip value.

#### Syntax

```
HRESULT SetClip (double clip);
```

#### Parameters

Name	Direction	Description
clip	in	The desired clip value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.7.23 IBMDSwitcherTransitionDVEParameters::GetGain method

The **GetGain** method returns the current clip.

### Syntax

```
HRESULT GetGain (double* gain);
```

### Parameters

Name	Direction	Description
gain	out	The current gain.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

## 3.2.7.24 IBMDSwitcherTransitionDVEParameters::SetGain method

The **SetGain** method sets the gain.

### Syntax

```
HRESULT SetGain (double gain);
```

### Parameters

Name	Direction	Description
gain	in	The desired gain.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 3.2.7.25 IBMDSwitcherTransitionDVEParameters::GetInverse method

The **GetInverse** method returns the current inverse flag.

#### Syntax

```
HRESULT      GetInverse (boolean* inverse);
```

#### Parameters

Name	Direction	Description
inverse	out	The current inverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

### 3.2.7.26 IBMDSwitcherTransitionDVEParameters::SetInverse method

The **SetInverse** method sets the inverse flag.

#### Syntax

```
HRESULT      SetInverse (boolean inverse);
```

#### Parameters

Name	Direction	Description
inverse	in	The desired inverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 3.2.7.27 IBMDSwitcherTransitionDVEParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionDVEParameters** object. Pass an object implementing the **IBMDSwitcherTransitionDVEParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT          AddCallback (IBMDSwitcherTransitionDVEParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionDVEParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.7.28 IBMDSwitcherTransitionDVEParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherTransitionDVEParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionDVEParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

3.2.8

IBMDSwitcherTransitionDVEParametersCallback Interface

The **IBMDSwitcherTransitionDVEParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionDVEParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionDVEParametersCallback	IID_IBMDSwitcherTransitionDVEParameters	An <b>IBMDSwitcherTransitionDVEParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionDVEParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionDVEParameters::RemoveCallback</b>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

### 3.2.8.1 IBMDSwitcherTransitionDVEParametersCallback::Notify method

The **Notify** method is called when **IBMDSwitcherTransitionDVEParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherTransitionDVEParametersEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionDVEParametersEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

3.2.9 **IBMDSwitcherTransitionStingerParameters Interface**

The **IBMDSwitcherTransitionStingerParameters** object interface is used for manipulating transition settings specific to stinger parameters.

**Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	An <b>IBMDSwitcherTransitionStingerParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

Public Member Functions	
Method	Description
GetSource	Get the current source.
SetSource	Set the source.
GetPreMultiplied	Get the current pre-multiplied flag.
SetPreMultiplied	Set the pre-multiplied flag.
GetClip	Get the current clip value.
SetClip	Set the clip value.
GetGain	Get the current gain.
SetGain	Set the gain.
GetInverse	Get the current inverse flag.
SetInverse	Set the inverse flag.
GetPreroll	Get the current pre-roll.
SetPreroll	Set the pre-roll.
GetClipDuration	Get the current clip duration.
SetClipDuration	Set the clip duration.



# SECTION 3

## Advanced Transitions

Public Member Functions	
Method	Description
GetTriggerPoint	Get the current trigger point.
SetTriggerPoint	Set the trigger point.
GetMixRate	Get the current mix rate.
SetMixRate	Set the mix rate.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

# SECTION 3 Advanced Transitions

## 3.2.9.1 IBMDSwitcherTransitionStingerParameters::GetSource method

The **GetSource** method returns the current source.

### Syntax

```
HRESULT GetSource (BMDSwitcherStingerTransitionSource* src);
```

### Parameters

Name	Direction	Description
src	out	The current source.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The src parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.9.2 IBMDSwitcherTransitionStingerParameters::SetSource method

The **SetSource** method sets the rate in frames.

### Syntax

```
HRESULT SetSource (BMDSwitcherStingerTransitionSource src);
```

### Parameters

Name	Direction	Description
src	in	The desired source.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The src parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.9.3 IBMDSwitcherTransitionStingerParameters::GetPreMultiplied method

The **GetPreMultiplied** method returns the current pre-multiplied flag.

### Syntax

```
HRESULT GetPreMultiplied (boolean* preMultiplied);
```

### Parameters

Name	Direction	Description
preMultiplied	out	The current pre-multiplied flag.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The preMultiplied parameter is invalid.

## 3.2.9.4 IBMDSwitcherTransitionStingerParameters::SetPreMultiplied method

The **SetPreMultiplied** method sets the pre-multiplied flag.

### Syntax

```
HRESULT SetPreMultiplied (boolean preMultiplied);
```

### Parameters

Name	Direction	Description
preMultiplied	in	The desired pre-multiplied flag.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.9.5 IBMDSwitcherTransitionStingerParameters::GetClip method

The **GetClip** method returns the current clip value.

#### Syntax

```
HRESULT GetClip (double* clip);
```

#### Parameters

Name	Direction	Description
clip	out	The current clip value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The clip parameter is invalid.

### 3.2.9.6 IBMDSwitcherTransitionStingerParameters::SetClip method

The **SetClip** method sets the clip value.

#### Syntax

```
HRESULT SetClip (double clip);
```

#### Parameters

Name	Direction	Description
clip	in	The desired clip value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.9.7 IBMDSwitcherTransitionStingerParameters::GetGain method

The **GetGain** method returns the current clip.

#### Syntax

```
HRESULT GetGain (double* gain);
```

#### Parameters

Name	Direction	Description
gain	out	The current gain.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

### 3.2.9.8 IBMDSwitcherTransitionStingerParameters::SetGain method

The **SetGain** method sets the gain.

#### Syntax

```
HRESULT SetGain (double gain);
```

#### Parameters

Name	Direction	Description
gain	in	The desired gain.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## SECTION 3 Advanced Transitions

### 3.2.9.9 IBMDSwitcherTransitionStingerParameters::GetInverse method

The **GetInverse** method returns the current inverse flag.

#### Syntax

```
HRESULT GetInverse (boolean* inverse);
```

#### Parameters

Name	Direction	Description
inverse	out	The current inverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

### 3.2.9.10 IBMDSwitcherTransitionStingerParameters::SetInverse method

The **SetInverse** method sets the inverse flag.

#### Syntax

```
HRESULT SetInverse (boolean inverse);
```

#### Parameters

Name	Direction	Description
inverse	in	The desired inverse flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.9.11 IBMDSwitcherTransitionStingerParameters::GetPreroll method

The **GetPreroll** method returns the current pre-roll.

### Syntax

```
HRESULT GetPreroll (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	out	The current pre-roll in frames.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

## 3.2.9.12 IBMDSwitcherTransitionStingerParameters::SetPreroll method

The **SetPreroll** method sets the pre-roll.

### Syntax

```
HRESULT SetPreroll (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired pre-roll in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



# SECTION 3 Advanced Transitions

## 3.2.9.13 IBMDSwitcherTransitionStingerParameters::GetClipDuration method

The **GetClipDuration** method returns the current clip duration.

### Syntax

```
HRESULT GetClipDuration (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	out	The current clip duration in frames.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

## 3.2.9.14 IBMDSwitcherTransitionStingerParameters::SetClipDuration method

The **SetClipDuration** method sets the clip duration.

### Syntax

```
HRESULT SetClipDuration (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired clip duration in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.9.15 IBMDSwitcherTransitionStingerParameters::GetTriggerPoint method

The **GetTriggerPoint** method returns the current trigger point.

### Syntax

```
HRESULT GetTriggerPoint (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	out	The current trigger point in frames.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

## 3.2.9.16 IBMDSwitcherTransitionStingerParameters::SetTriggerPoint method

The **SetTriggerPoint** method sets the trigger point.

### Syntax

```
HRESULT SetTriggerPoint (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired trigger point in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.9.17 IBMDSwitcherTransitionStingerParameters::GetMixRate method

The **GetMixRate** method returns the current mix rate.

### Syntax

```
HRESULT GetMixRate (uint32_t* frames);
```

### Parameters

Name	Direction	Description
frames	out	The current mix rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

## 3.2.9.18 IBMDSwitcherTransitionStingerParameters::SetMixRate method

The **SetMixRate** method sets the mix rate.

### Syntax

```
HRESULT SetMixRate (uint32_t frames);
```

### Parameters

Name	Direction	Description
frames	in	The desired mix rate in frames.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 3.2.9.19 IBMDSwitcherTransitionStingerParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionStingerParameters** object. Pass an object implementing the **IBMDSwitcherTransitionStingerParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherTransitionStingerParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionStingerParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.9.20 IBMDSwitcherTransitionStingerParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTransitionStingerParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionStingerParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

3.2.10

IBMDSwitcherTransitionStingerParametersCallback Interface

The **IBMDSwitcherTransitionStingerParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionStingerParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionStingerParameters	IID_IBMDSwitcherTransitionStingerParameters	An <b>IBMDSwitcherTransitionStingerParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionStingerParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionStingerParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

3.2.10.1 **IBMDSwitcherTransitionStingerParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherTransitionStingerParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

**Syntax**

**HRESULT** Notify (BMDSwitcherTransitionStingerParametersEventType eventType);

**Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionStingerParametersEventType</b> that describes the type of event that has occurred.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

# SECTION 3 Advanced Transitions

## 3.2.11 IBMDSwitcherTransitionParameters Interface

The IBMDSwitcherTransitionParameters object interface is used for manipulating transition settings specific to Stinger parameters.

### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionParameters	IID_IBMDSwitcherTransitionParameters	An <b>IBMDSwitcherTransitionParameters</b> object interface can be obtained with <b>IBMDSwitcherMixEffectBlock::QueryInterface</b> .

### Public Member Functions

Method	Description
GetTransitionStyle	Get the current transition style.
GetNextTransitionStyle	Get the next transition style.
SetNextTransitionStyle	Set the next transition style.
GetTransitionSelection	Get the current transition selection.
SetNextTransitionSelection	Set the next transition selection.
GetNextTransitionSelection	Get the next transition selection.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.



# SECTION 3 Advanced Transitions

## 3.2.11.1 IBMDSwitcherTransitionParameters::GetTransitionStyle method

The **GetTransitionStyle** method returns the current transition style.

### Syntax

```
HRESULT GetTransitionStyle (BMDSwitcherTransitionStyle* style);
```

### Parameters

Name	Direction	Description
style	out	The current transition style.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The style parameter is invalid.

## 3.2.11.2 IBMDSwitcherTransitionParameters::GetNextTransitionStyle method

The **GetNextTransitionStyle** method returns the next transition style.

### Syntax

```
HRESULT GetNextTransitionStyle (BMDSwitcherTransitionStyle* style);
```

### Parameters

Name	Direction	Description
style	out	The next transition style.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The style parameter is invalid.

# SECTION 3 Advanced Transitions

## 3.2.11.3 IBMDSwitcherTransitionParameters::SetNextTransitionStyle method

The **SetNextTransitionStyle** method sets the rate in frames.

### Syntax

```
HRESULT SetNextTransitionStyle (BMDSwitcherTransitionStyle style);
```

### Parameters

Name	Direction	Description
style	in	The desired style.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The style parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.11.4 IBMDSwitcherTransitionParameters::GetTransitionSelection method

The **GetTransitionSelection** method returns the current transition selection.

#### Syntax

```
HRESULT GetTransitionSelection (BMDSwitcherTransitionSelection* selection);
```

#### Parameters

Name	Direction	Description
selection	out	The current transition selection.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The selection parameter is invalid.

SECTION

3

Advanced Transitions

3.2.11.5

IBMDSwitcherTransitionParameters::SetNextTransitionSelection method

The **SetNextTransitionSelection** method sets the next transition selection.

**Syntax**

**HRESULT**           SetNextTransitionSelection (BMDSwitcherTransitionSelection selection);

**Parameters**

Name	Direction	Description
selection	in	The desired next transition selection.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The selection parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.11.6 IBMDSwitcherTransitionParameters::GetNextTransitionSelection method

The **GetNextTransitionSelection** method returns the next transition selection.

#### Syntax

```
HRESULT GetNextTransitionSelection (BMDSwitcherTransitionSelection* selection);
```

#### Parameters

Name	Direction	Description
selection	out	The next transition selection.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The selection parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.11.7 IBMDSwitcherTransitionParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherTransitionParameters** object. Pass an object implementing the **IBMDSwitcherTransitionParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherTransitionParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

# SECTION 3

## Advanced Transitions

### 3.2.11.8 IBMDSwitcherTransitionParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTransitionParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTransitionParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

3.2.12

IBMDSwitcherTransitionParametersCallback Interface

The **IBMDSwitcherTransitionParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherTransitionParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTransitionParameters	IID_IBMDSwitcherTransitionParameters	An <b>IBMDSwitcherTransitionParametersCallback</b> object interface is installed with <b>IBMDSwitcherTransitionParameters::AddCallback</b> and removed with <b>IBMDSwitcherTransitionParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.



3.2.12.1 **IBMDSwitcherTransitionParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherTransitionParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

**Syntax**

```
HRESULT Notify (BMDSwitcherTransitionParametersEventType eventType);
```

**Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherTransitionParametersEventType</b> that describes the type of event that has occurred.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 4 Switcher Media

All media used by the switcher comes from its media pool, which consists of still images or video clips. Developers can upload and download from the media pool of a switcher using this SDK.

### 4.1 General Information

#### 4.1.1 Uploading a Still or Clip

Here are the basic steps of uploading media to a switcher:

1. Ensure you are connected to a switcher and have a switcher object. Please refer to the Basic Switcher Control section for how to do this.
2. Get the **IBMDSwitcherMediaPool** interface from the switcher object, and use **IBMDSwitcherMediaPool::GetStills** to get the interface dedicated to all stills or **IBMDSwitcherMediaPool::GetClip** to get the interface dedicated to a particular clip.
3. **IBMDSwitcherStills::Lock** and **IBMDSwitcherClip::Lock** requests a lock of the switcher's stills/clip and the **IBMDSwitcherLockCallback** interface should then be used to be informed of when a lock is obtained. Many media pool operations require you to have a lock first.
4. If you are transferring to a clip then you probably want to stop users playing/downloading any frame in the clip by calling **IBMDSwitcherClip::SetInvalid**.
5. Use **IBMDSwitcherMediaPool::CreateFrame** to generate a frame object that will eventually be passed to the upload system. Populate this with your image data by filling in the frame's buffer, which is available via **IBMDSwitcherFrame::GetBytes**. Note that you do not need a lock to create a frame, but it is important that the chosen dimensions for the frame match those of the switcher's video mode when you proceed with the upload step.
6. Call **IBMDSwitcherStill::Upload** or **IBMDSwitcherClip::UploadFrame** to begin the transfer of the frame to the switcher. You will be notified of the outcome of this process by the **IBMDSwitcherStillsCallback** or **IBMDSwitcherClipCallback** interfaces. Regardless of outcome, this notification will also include the frame that was sent.
7. If you are transferring to a clip then you may want to repeatedly perform steps 4 and 5, although only one frame is permitted to be transferred at a time, and upon completing clip transfers you need to use **IBMDSwitcherClip::SetValid**.

## SECTION 4 Switcher Media

8. Unlock the stills or clip. If you are uploading multiple stills then you may want to repeatedly lock and unlock the stills pool to allow other users to obtain a lock.

### 4.1.2 Downloading a Still or Clip

The steps are very similar to uploading:

9. Ensure you are connected to a switcher and have a switcher object. Please refer to the Basic Switcher Control section for how to do this.
10. Get the **IBMDSwitcherMediaPool** interface from the switcher object, and use **IBMDSwitcherMediaPool::GetStills** to get the interface dedicated to all stills or **IBMDSwitcherMediaPool::GetClip** to get the interface dedicated to a particular clip.
11. **IBMDSwitcherStills::Lock** and **IBMDSwitcherClip::Lock** requests a lock of the switcher's stills/clip and the **IBMDSwitcherLockCallback** interface should then be used to be informed of when a lock is obtained. Many media pool operations require you to have a lock first.
12. Call **IBMDSwitcherStill::Download** or **IBMDSwitcherClip::DownloadFrame** to begin the transfer of a frame from the switcher. You will be notified of the outcome of this process by the **IBMDSwitcherStillsCallback** or **IBMDSwitcherClipCallback** interfaces. For successful downloads, this notification will also include the frame that was requested.
13. Unlock the stills or clip. If you are downloading multiple stills then you may want to repeatedly lock and unlock the stills pool to allow other users to obtain a lock.

# SECTION 4 Switcher Media

## 4.2 Media Data Types

### 4.2.1 Switcher Pixel Format

**BMDSwitcherPixelFormat** enumerates the possible pixel formats for `IBMDSwitcherFrame`.

<b>bmdSwitcherPixelFormat8BitARGB</b>	Four bytes per pixel, alpha, red, green, blue.
<b>bmdSwitcherPixelFormat8BitXRGB</b>	Four bytes per pixel, padding, red, green, blue.
<b>bmdSwitcherPixelFormat8BitYUV</b>	Four bytes per two pixels, cb, y0, cr, y1.
<b>bmdSwitcherPixelFormat10BitYUVA</b>	Eight bytes per two pixels, a0, cb, y0, a1, cr, y1.

### 4.2.2 Media Player Source Type

**BMDSwitcherMediaPlayerSourceType** enumerates the possible source types for `IBMDSwitcherMediaPlayer`.

<b>bmdSwitcherMediaPlayerSourceTypeStill</b>	Still.
<b>bmdSwitcherMediaPlayerSourceTypeClip</b>	Clip.

### 4.2.3 Media Pool Event Type

**BMDSwitcherMediaPoolEventType** enumerates the possible event types for `IBMDSwitcherStillsCallback` and `IBMDSwitcherClipCallback`.

<b>bmdSwitcherMediaPoolEventTypeValidChanged</b>	The validity has changed.
<b>bmdSwitcherMediaPoolEventTypeNameChanged</b>	The name has changed.
<b>bmdSwitcherMediaPoolEventTypeHashChanged</b>	The hash has changed.
<b>bmdSwitcherMediaPoolEventTypeAudioValidChanged</b>	The audio validity has changed.
<b>bmdSwitcherMediaPoolEventTypeLockBusy</b>	All clients receive this when any client obtains a lock.
<b>bmdSwitcherMediaPoolEventTypeLockIdle</b>	All clients receive this when no client has lock.
<b>bmdSwitcherMediaPoolEventTypeTransferCompleted</b>	The transfer has completed.
<b>bmdSwitcherMediaPoolEventTypeTransferCancelled</b>	The transfer has cancelled.
<b>bmdSwitcherMediaPoolEventTypeTransferFailed</b>	The transfer has failed.
<b>bmdSwitcherMediaPoolEventTypeAudioNameChanged</b>	The audio name has changed.
<b>bmdSwitcherMediaPoolEventTypeAudioHashChanged</b>	The audio hash has changed.

4.3

Interface Reference

4.3.1

IBMDSwitcherMediaPlayerCallback Interface

The **IBMDSwitcherMediaPlayerCallback** object interface is a callback class containing methods that are called when the source, state or properties change for an **IBMDSwitcherMediaPlayer** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPlayer	IID_IBMDSwitcherMediaPlayer	An <b>IBMDSwitcherMediaPlayerCallback</b> object interface is installed with <b>IBMDSwitcherMediaPlayer::AddCallback</b> and removed with <b>IBMDSwitcherMediaPlayer::RemoveCallback</b>

Public Member Functions	
Method	Description
SourceChanged	Called when the source changes.
PlayingChanged	Called when playing changes.
LoopChanged	Called when loop changes.
AtBeginningChanged	Called when the current clip frame index changes to or from zero.
ClipFrameChanged	Called when the clip frame index is set.

# SECTION 4 Switcher Media

## 4.3.1.1 IBMDSwitcherMediaPlayerCallback::SourceChanged method

The **SourceChanged** method is called when the media player source changes.

### Syntax

```
HRESULT      SourceChanged (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

## 4.3.1.2 IBMDSwitcherMediaPlayerCallback::PlayingChanged method

The **PlayingChanged** method is called when the media player playing state changes.

### Syntax

```
HRESULT      PlayingChanged (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

# SECTION 4 Switcher Media

## 4.3.1.3 IBMDSwitcherMediaPlayerCallback::LoopChanged method

The **LoopChanged** method is called when the media player loop property changes.

### Syntax

**HRESULT** LoopChanged (void);

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

## 4.3.1.4 IBMDSwitcherMediaPlayerCallback::AtBeginningChanged method

The **AtBeginningChanged** method is called when the media player current clip frame index changes to or from zero.

### Syntax

**HRESULT** AtBeginningChanged (void);

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

# SECTION 4 Switcher Media

## 4.3.1.5 IBMDSwitcherMediaPlayerCallback::ClipFrameChanged method

The **ClipFrameChanged** method is called when the media player clip frame index is set.

### Syntax

```
HRESULT ClipFrameChanged (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.



4.3.2 IBMDSwitcherMediaPlayerIterator Interface

The **IBMDSwitcherMediaPlayerIterator** is used to enumerate the available media players.

A reference to an **IBMDSwitcherMediaPlayerIterator** object interface may be obtained from an **IBMDSwitcher** object interface using the **CreateIterator** method. Pass **IID\_IBMDSwitcherMediaPlayerIterator** for the iid parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher:: CreateIterator</b> can return an <b>IBMDSwitcherMediaPlayerIterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherMediaPlayer</b> object interface.

#### 4.3.2.1 IBMDSwitcherMediaPlayerIterator::Next method

The Next method returns the next available **IBMDSwitcherMediaPlayer** object interface.

##### Syntax

```
HRESULT         Next (IBMDSwitcherMediaPlayer** mediaPlayer);
```

##### Parameters

Name	Direction	Description
mediaPlayer	out	<b>IBMDSwitcherMediaPlayer</b> object interface or NULL when no more media players are available.

##### Return Values

Value	Description
E_FALSE	Failure.
S_OK	Success.
E_POINTER	The mediaPlayer parameter is invalid.

### 4.3.3 IBMDSwitcherMediaPlayer Interface

The **IBMDSwitcherMediaPlayer** object interface provides the ability to play stills and clips sourced from the media pool.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPlayerIterator	IID_IBMDSwitcherMediaPlayerIterator	<b>IBMDSwitcherMediaPlayerIterator::Next</b> returns an <b>IBMDSwitcherMediaPlayer</b> object interface for each available media player.

Public Member Functions	
Method	Description
GetSource	Gets the media player source.
SetSource	Sets the media player source.
GetPlaying	Gets the media player playing state.
SetPlaying	Sets the media player playing state.
GetLoop	Gets the media player loop property.
SetLoop	Sets the media player loop property.
GetAtBeginning	Gets the media player at beginning state.
SetAtBeginning	Sets the media player at beginning state.
GetClipFrame	Gets the media player clip frame index.
SetClipFrame	Sets the media player clip frame index.
AddCallback	Adds a media player callback.
RemoveCallback	Removes a media player callback.

#### 4.3.3.1 IBMDSwitcherMediaPlayer::GetSource method

The **GetSource** method gets the source type and index for the media player.

##### Syntax

```
HRESULT      GetSource (BMDSwitcherMediaPlayerSourceType* type, uint32_t* index);
```

##### Parameters

Name	Direction	Description
type	out	<b>BMDSwitcherMediaPlayerSourceType</b> specifying the source as a still or clip.
index	out	Integer specifying the index of the source.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The type or index parameter is invalid.

#### 4.3.3.2 IBMDSwitcherMediaPlayer::SetSource method

The **SetSource** method sets the source type and index for the media player.

##### Syntax

```
HRESULT      SetSource (BMDSwitcherMediaPlayerSourceType type, uint32_t index);
```

##### Parameters

Name	Direction	Description
type	in	<b>BMDSwitcherMediaPlayerSourceType</b> specifying the source as a still or clip.
index	in	Integer specifying the index of the source.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	The type or index parameter is invalid.

## 4.3.3.3 IBMDSwitcherMediaPlayer::GetPlaying method

The **GetPlaying** method gets the playing state for the media player.

### Syntax

```
HRESULT      GetPlaying (boolean* playing);
```

### Parameters

Name	Direction	Description
playing	out	Boolean value specifying the playing state.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The playing parameter is invalid.

## 4.3.3.4 IBMDSwitcherMediaPlayer::SetPlaying method

The **SetPlaying** method sets the playing state for the media player.

### Syntax

```
HRESULT      SetPlaying (boolean playing);
```

### Parameters

Name	Direction	Description
playing	in	Boolean value specifying the playing state.

### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

## 4.3.3.5 IBMDSwitcherMediaPlayer::GetLoop method

The **GetLoop** method gets the loop property for the media player.

### Syntax

```
HRESULT GetLoop (boolean* loop);
```

### Parameters

Name	Direction	Description
loop	out	Boolean value specifying the loop property.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The loop parameter is invalid.

## 4.3.3.6 IBMDSwitcherMediaPlayer::SetLoop method

The **SetLoop** method sets the loop property for the media player.

### Syntax

```
HRESULT SetLoop (boolean loop);
```

### Parameters

Name	Direction	Description
loop	in	Boolean value specifying the loop property.

### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 4.3.3.7 IBMDSwitcherMediaPlayer::GetAtBeginning method

The **GetAtBeginning** method gets the at beginning property for the media player.

##### Syntax

```
HRESULT      GetAtBeginning (boolean* atBeginning);
```

##### Parameters

Name	Direction	Description
atBeginning	out	Boolean value that is true when the current frame index is zero and false otherwise.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The atBeginning parameter is invalid.



## 4.3.3.8 IBMDSwitcherMediaPlayer::SetAtBeginning method

The **SetAtBeginning** method sets the current frame index to zero for the media player.

### Syntax

```
HRESULT SetAtBeginning ();
```

### Parameters

none.

### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

## 4.3.3.9 IBMDSwitcherMediaPlayer::GetClipFrame method

The **GetClipFrame** method gets the clip frame index for the media player when it is not playing.

### Syntax

```
HRESULT GetClipFrame (uint32_t* clipFrameIndex);
```

### Parameters

Name	Direction	Description
clipFrameIndex	out	Integer value specifying the clip frame index.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The clipFrameIndex parameter is invalid.

# SECTION 4 Switcher Media

## 4.3.3.10 IBMDSwitcherMediaPlayer::SetClipFrame method

The **SetClipFrame** method sets the clip frame index for the media player if it is not playing.

### Syntax

```
HRESULT SetClipFrame (uint32_t clipFrameIndex);
```

### Parameters

Name	Direction	Description
clipFrameIndex	in	Integer value specifying the clip frame index.

### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 4.3.3.11 IBMDSwitcherMediaPlayer::AddCallback method

The **AddCallback** method configures a callback to be called when the properties change for an **IBMDSwitcherMediaPlayer** object. Pass an object implementing the **IBMDSwitcherMediaPlayerCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

**HRESULT**           AddCallback (IBMDSwitcherMediaPlayerCallback\* callback);

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMediaPlayerCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 4.3.3.12 IBMDSwitcherMediaPlayer::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT AddCallback (IBMDSwitcherMediaPlayerCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMediaPlayerCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 4.3.4 IBMDSwitcherFrame Interface

The **IBMDSwitcherFrame** object interface represents a frame and provides access to the frame’s buffer and frame properties.

##### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPool	IID_IBMDSwitcherMediaPool	<b>IBMDSwitcherMediaPool::CreateFrame</b> returns an <b>IBMDSwitcherFrame</b> object.

Public Member Functions	
Method	Description
GetWidth	Gets the frame width in pixels.
GetHeight	Gets the frame height in pixels.
GetRowBytes	Gets the frame row size in bytes.
GetPixelFormat	Gets the pixel format.
GetBytes	Gets a pointer to the frame’s buffer.

## 4.3.4.1 IBMDSwitcherFrame::GetWidth method

The **GetWidth** method returns the width of the frame in pixels.

### Syntax

```
int32 _t      GetWidth (void);
```

### Parameters

none.

### Return Values

Value	Description
Width	Frame width in pixels.

## 4.3.4.2 IBMDSwitcherFrame::GetHeight method

The **GetHeight** method returns the height of the frame in pixels.

### Syntax

```
int32 _t      GetHeight (void);
```

### Parameters

none.

### Return Values

Value	Description
Height	Frame height in pixels.

## 4.3.4.3 IBMDSwitcherFrame::GetRowBytes method

The **GetRowBytes** method returns the number of bytes per row in the frame.

### Syntax

```
int32_t GetRowBytes (void);
```

### Parameters

none.

### Return Values

Value	Description
ByteCount	Frame row size in bytes.

## 4.3.4.4 IBMDSwitcherFrame::GetPixelFormat method

The **GetPixelFormat** method returns the pixel format of the frame.

### Syntax

```
BMDSwitcherPixelFormat GetPixelFormat (void);
```

### Parameters

none.

### Return Values

Value	Description
PixelFormat	Frame pixel format.

## 4.3.4.5 IBMDSwitcherFrame::GetBytes method

The **GetBytes** method allows direct access to the data buffer of the frame.  
The audio format is raw 24 bit, 2 channel, 48 khz.

### Syntax

```
HRESULT GetBytes (void** buffer);
```

### Parameters

Name	Direction	Description
buffer	out	Pointer to the frame's raw buffer – only valid while while object remains valid.

### Return Values

Value	Description
E_POINTER	The buffer parameter is invalid.
S_OK	Success.



# SECTION 4 Switcher Media

## 4.3.5 IBMDSwitcherAudio Interface

The **IBMDSwitcherAudio** object interface represents audio and provides access to the audio's buffer and audio size.

### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPool	IID_IBMDSwitcherMediaPool	<b>IBMDSwitcherMediaPool::CreateAudio</b> returns an <b>IBMDSwitcherAudio</b> object.

### Public Member Functions

Method	Description
GetSize	Gets the audio size in bytes.
GetBytes	Gets the audio buffer pointer.

## 4.3.5.1 IBMDSwitcherFrame::GetSize method

The **GetSize** method returns the size of the audio in bytes.

### Syntax

```
int32_t      GetSize (void);
```

### Parameters

none.

### Return Values

Value	Description
ByteCount	Audio size in bytes.

## 4.3.5.2 IBMDSwitcherAudio::GetBytes method

The **GetBytes** method allows direct access to the data buffer of the audio.

### Syntax

```
HRESULT      GetBytes (void** buffer);
```

### Parameters

Name	Direction	Description
buffer	out	Pointer to the audio's raw buffer – only valid while object remains valid.

### Return Values

Value	Description
E_POINTER	The buffer parameter is invalid.
S_OK	Success.

### 4.3.6 IBMDSwitcherLockCallback Interface

The **IBMDSwitcherLockCallback** object interface is a callback class with an **Obtained** method that is called when the client receives a lock. Like all callback methods, **Obtained** may be called from another thread.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherStills	IID_IBMDSwitcherStills	An <b>IBMDSwitcherLockCallback</b> object interface is installed with <b>IBMDSwitcherStills::Lock</b> and removed with <b>IBMDSwitcherStills::Unlock</b>
IBMDSwitcherClip	IID_IBMDSwitcherClip	An <b>IBMDSwitcherLockCallback</b> object interface is installed with <b>IBMDSwitcherClip::Lock</b> and removed with <b>IBMDSwitcherClip::Unlock</b>

Public Member Functions	
Method	Description
Obtained	Called when the client receives a lock.

# SECTION 4 Switcher Media

## 4.3.6.1 IBMDSwitcherLockCallback::Obtained method

The **Obtained** method is called only for the client that receives the lock.

### Syntax

```
HRESULT      Obtained (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

4.3.7 IBMDSwitcherStillsCallback Interface

The **IBMDSwitcherStillsCallback** object interface is a callback class with a **Notify** method that is called when an event occurs for an **IBMDSwitcherStills** object. Like all callback methods, **Notify** may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherStills	IID_IBMDSwitcherStills	An <b>IBMDSwitcherStillsCallback</b> object interface is installed with <b>IBMDSwitcherStills::AddCallback</b> and removed with <b>IBMDSwitcherStills::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an stills event occurs.

### 4.3.7.1 IBMDSwitcherStillsCallback::Notify method

The **Notify** method is called when a stills event occurs. See **BMDSwitcherMediaPoolEventType** for a list of event types that may occur.

<b>bmdSwitcherMediaPoolEventTypeTransferCompleted</b>	<b>IBMDSwitcherStills::Upload</b> and <b>IBMDSwitcherStills::Download</b> .
<b>bmdSwitcherMediaPoolEventTypeTransferCancelled</b>	<b>IBMDSwitcherStills::Upload</b> only.
<b>bmdSwitcherMediaPoolEventTypeTransferFailed</b>	<b>BMDSwitcherStills::Upload</b> only.

**IBMDSwitcherFrame ::AddRef** must be called on the frame to extend its lifetime beyond the scope of this method.

#### Syntax

**HRESULT**            Notify (BMDSwitcherMediaPoolEventType eventType, IBMDSwitcherFrame\* frame, int32\_t index);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherMediaPoolEventType</b> that describes the type of event that has occurred.
frame	in	The IBMDSwitcherFrame that is being transferred. May be <b>NULL</b> .
index	in	Specifies the still for the eventType. The index is -1 when eventType is not specific to an individual still.

#### Return Values

Value	Description
S_OK	Success.

### 4.3.8 IBMDSwitcherStills Interface

The **IBMDSwitcherStills** object interface represents the media pool stills.

The switcher stills interface provides methods to transfer stills and change still properties.  
No more than one transfer can occur at a time.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPool	IID_IBMDSwitcherMediaPool	<b>IBMDSwitcherMediaPool::GetStills</b> returns an <b>IBMDSwitcherStills</b> object.

Public Member Functions	
Method	Description
GetCount	Gets the number of stills.
IsValid	Gets the validity of a still.
GetName	Gets the name of a still.
GetHash	Gets the hash of a still.
SetInvalid	Invalidates a still.
Lock	Locks all stills.
Unlock	Unlocks all stills.
Upload	Uploads a still.
Download	Downloads a still.
CancelTransfer	Cancels the upload or download.
GetProgress	Gets the transfer progress.
AddCallback	Adds a stills callback.
RemoveCallback	Removes a stills callback.

## 4.3.8.1 IBMDSwitcherStills::GetCount method

The **GetCount** method returns the number of stills.

### Syntax

```
HRESULT GetCount (uint32_t* count);
```

### Parameters

Name	Direction	Description
count	out	Number of stills.

### Return Values

Value	Description
E_POINTER	The count parameter is NULL.
S_OK	Success.

## 4.3.8.2 IBMDSwitcherStills::IsValid method

The **IsValid** method returns the validity of a still. A valid still can be downloaded and used by the media player.

### Syntax

```
HRESULT IsValid (uint32_t index, bool* valid);
```

### Parameters

Name	Direction	Description
index	in	Still index.
valid	out	Validity of the still.

### Return Values

Value	Description
E_POINTER	The valid parameter is NULL.
S_OK	Success.



#### 4.3.8.3 IBMDSwitcherStills::GetName method

The **GetName** method returns the name of a still.

##### Syntax

```
HRESULT      GetName (uint32_t index, string* name);
```

##### Parameters

Name	Direction	Description
index	in	Still index.
name	out	Still name.

##### Return Values

Value	Description
E_POINTER	The name parameter is NULL.
S_OK	Success.

#### 4.3.8.4 IBMDSwitcherStills::SetName method

The **SetName** method sets the name of a still.

##### Syntax

```
HRESULT      SetName (uint32_t index, string name);
```

##### Parameters

Name	Direction	Description
index	in	Still index.
name	in	The still name.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The name parameter is invalid.
E_INVALIDARG	The index parameter is invalid.

#### 4.3.8.5 IBMDSwitcherStills::GetHash method

The **GetHash** method returns the hash of a still.

##### Syntax

```
HRESULT      GetHash (uint32_t index, BMDSwitcherHash* hash);
```

##### Parameters

Name	Direction	Description
index	in	Still index.
hash	out	Still hash.

##### Return Values

Value	Description
E_POINTER	The hash parameter is NULL.
S_OK	Success.

#### 4.3.8.6 IBMDSwitcherStills::SetInvalid method

The **SetInvalid** method invalidates a still for all switcher users. A still is set valid after a successful upload. This method will only be successful if you have a lock or no other connected client has a lock.

##### Syntax

```
HRESULT          SetInvalid (uint32_t index);
```

##### Parameters

Name	Direction	Description
index	in	Still index.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 4.3.8.7 IBMDSwitcherStills::Lock method

The **Lock** method obtains a client lock for stills. Pass an object implementing the **IBMDSwitcherLockCallback** interface to receive **IBMDSwitcherLockCallback::Obtained** when the client obtains the stills lock.

##### Syntax

```
HRESULT Lock (IBMDSwitcherLockCallback* lockCallback);
```

##### Parameters

Name	Direction	Description
lockCallback	in	Callback object implementing the <b>IBMDSwitcherLockCallback</b> object interface.

##### Return Values

Value	Description
E_POINTER	The lockCallback parameter is invalid.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.8.8 IBMDSwitcherStills::Unlock method

The **Unlock** method releases the previous client lock for stills.

##### Syntax

```
HRESULT          Unlock (IBMDSwitcherLockCallback* lockCallback);
```

##### Parameters

Name	Direction	Description
lockCallback	in	Callback object implementing the <b>IBMDSwitcherLockCallback</b> object interface.

##### Return Values

Value	Description
E_POINTER	The lockCallback parameter is invalid.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.8.9 IBMDSwitcherStills::Upload method

The **Upload** method transfers a still to the media pool. The client must hold the stills lock for the duration of the transfer. No more than one transfer can occur at a time. Do not write to the frame’s buffer during the transfer.

##### Syntax

```
HRESULT Upload (uint32_t index, string* name, IBMDSwitcherFrame* frame);
```

##### Parameters

Name	Direction	Description
<b>index</b>	in	Destination still index.
<b>name</b>	in	Destination still name.
<b>frame</b>	in	Still frame to upload. The frame dimensions must match the switcher video mode.

##### Return Values

Value	Description
E_POINTER	The lockCallback parameter is invalid.
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	The index parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the stills lock.

#### 4.3.8.10 IBMDSwitcherStills::Download method

The **Download** method transfers a still from the media pool. The client must hold the stills lock for the duration of the transfer. No more than one transfer can occur at a time.

##### Syntax

```
HRESULT          Download (uint32_t index);
```

##### Parameters

Name	Direction	Description
index	in	Index of still to download.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	The index parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the stills lock.



## 4.3.8.11 IBMDSwitcherStills::CancelTransfer method

The **CancelTransfer** method cancels the pending transfer. If there is no pending transfer then this method has no effect.

### Syntax

```
HRESULT          CancelTransfer ();
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

## 4.3.8.12 IBMDSwitcherStills::GetProgress method

The **GetProgress** method gets the progress of the pending transfer. If there is no pending transfer then progress is zero.

### Syntax

```
HRESULT          GetProgress (double* progress);
```

### Parameters

Name	Direction	Description
progress	out	Transfer progress. Range is between 0.0 and 1.0.

### Return Values

Value	Description
E_POINTER	The progress parameter is invalid.
S_OK	Success.

#### 4.3.8.13 IBMDSwitcherStills::AddCallback method

The **AddCallback** method configures a callback to be called when an event occurs for an **IBMDSwitcherStills** object. Pass an object implementing the **IBMDSwitcherStillsCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

**BMDSwitcherStillsCallback ::Notify** will be called immediately on the provided **IBMDSwitcherStillsCallback** callback object with one of the following **BMDSwitcherMediaPoolEventType** eventTypes:

**bmdSwitcherMediaPoolEventTypeLockBusy**  
**bmdSwitcherMediaPoolEventTypeLockIdle**

#### Syntax

```
HRESULT      AddCallback (IBMDSwitcherStillsCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherStillsCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 4.3.8.14 IBMDSwitcherStills::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherStillsCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherStillsCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

4.3.9

IBMDSwitcherClipCallback Interface

The **IBMDSwitcherClipCallback** object interface is a callback class with a **Notify** method that is called when an event occurs for an **IBMDSwitcherClip** object. Like all callback methods, **Notify** may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherClip	IID_IBMDSwitcherClip	An <b>IBMDSwitcherClipCallback</b> object interface is installed with <b>IBMDSwitcherClip::AddCallback</b> and removed with <b>IBMDSwitcherClip::RemoveCallback</b>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

## 4.3.9.1 IBMDSwitcherClipCallback::Notify method

The **Notify** method is called when a clip event occurs. See **BMDSwitcherMediaPoolEventType** for a list of event types that may occur.

The frame is set during the following transfer events, otherwise it is NULL:

<b>bmdSwitcherMediaPoolEventTypeTransferCompleted</b>	<b>IBMDSwitcherClip::Upload</b> and <b>IBMDSwitcherClip::Download</b> .
<b>bmdSwitcherMediaPoolEventTypeTransferCancelled</b>	<b>IBMDSwitcherClip::Upload</b> only.
<b>bmdSwitcherMediaPoolEventTypeTransferFailed</b>	<b>IBMDSwitcherClip::Upload</b> only.

The audio is set during the following transfer events, otherwise it is NULL:

<b>bmdSwitcherMediaPoolEventTypeTransferCompleted</b>	<b>IBMDSwitcherClip::UploadAudio</b> and <b>IBMDSwitcherClip::DownloadAudio</b> .
<b>bmdSwitcherMediaPoolEventTypeTransferCancelled</b>	<b>IBMDSwitcherClip::UploadAudio</b> only.
<b>bmdSwitcherMediaPoolEventTypeTransferFailed</b>	<b>IBMDSwitcherClip::UploadAudio</b> only.

**IBMDSwitcherFrame ::AddRef** must be called on the frame to extend its lifetime beyond the scope of this method.

**IBMDSwitcherAudio ::AddRef** must be called on the audio to extend its lifetime beyond the scope of this method.

### Syntax

```
HRESULT Notify (BMDSwitcherMediaPoolEventType eventType, IBMDSwitcherFrame* frame,
                int32_t frameIndex, IBMDSwitcherAudio* audio, int32_t clipIndex);
```

# SECTION 4 Switcher Media

## Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherMediaPoolEventType</b> that describes the type of event that has occurred.
frame	in	The <b>IBMDSwitcherFrame</b> that is being transferred. May be NULL.
frameIndex	in	Specifies the frame for the eventType. The index is -1 when eventType is not specific to an individual clip frame.
audio	in	The <b>IBMDSwitcherAudio</b> that is being transferred. May be NULL.
clipIndex	in	Specifies the clip for the eventType.

## Return Values

Value	Description
S_OK	Success.

### 4.3.10 IBMDSwitcherClip Interface

The **IBMDSwitcherClip** object interface represents the media pool clips.

The switcher clip interface provides methods to transfer clip frames and audio and to change clip properties. No more than one transfer can occur at a time.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPool	IID_IBMDSwitcherClip	<b>IBMDSwitcherMediaPool::GetClip</b> returns an <b>IBMDSwitcherClip</b> object.

Public Member Functions	
Method	Description
GetIndex	Gets the clip index.
IsValid	Gets the validity of the clip.
GetName	Gets the name of the clip.
SetValid	Sets the clip name and frame count and sets the clip valid.
SetInvalid	Invalidates the clip.
GetFrameCount	Gets the current clip frame count.
GetMaxFrameCount	Gets the maximum clip frame count.
IsFrameValid	Gets the validity of a clip frame.
GetFrameHash	Gets the hash of a clip frame.
IsAudioValid	Gets the validity of the clip audio.
GetAudioName	Gets the name of the clip audio.
GetAudioHash	Gets the hash of the clip audio.
SetAudioInvalid	Invalidates the clip audio.

# SECTION 4 Switcher Media

Public Member Functions	
Method	Description
Lock	Locks the clip.
Unlock	Unlocks the clip.
UploadFrame	Uploads a clip frame.
DownloadFrame	Downloads a clip frame.
UploadAudio	Uploads the clip audio.
DownloadAudio	Downloads the clip audio.
CancelTransfer	Cancels the transfer.
GetProgress	Gets the transfer progress.
AddCallback	Adds a clip callback.
RemoveCallback	Removes a clip callback.



## 4.3.10.1 IBMDSwitcherClip::GetIndex method

The **GetIndex** method returns the clip index.

### Syntax

```
HRESULT GetIndex (uint32_t* index);
```

### Parameters

Name	Direction	Description
index	out	The clip index.

### Return Values

Value	Description
S_OK	Success.

## 4.3.10.2 IBMDSwitcherClip::IsValid method

The **IsValid** method returns the validity of the clip. A valid clip can be downloaded and played by the media player.

### Syntax

```
HRESULT IsValid (bool* valid);
```

### Parameters

Name	Direction	Description
valid	out	Validity of the clip.

### Return Values

Value	Description
E_POINTER	The valid parameter is NULL.
S_OK	Success.

# SECTION 4 Switcher Media

## 4.3.10.3 IBMDSwitcherClip::GetName method

The **GetName** method returns the name of the clip.

### Syntax

```
HRESULT      GetName (string* name);
```

### Parameters

Name	Direction	Description
name	out	Clip name.

### Return Values

Value	Description
E_POINTER	The name parameter is NULL.
S_OK	Success.

#### 4.3.10.4 IBMDSwitcherClip::SetName method

The **SetName** method sets the name of the clip.

##### Syntax

```
HRESULT      SetName (string name);
```

##### Parameters

Name	Direction	Description
name	in	The clip name.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The name parameter is invalid.

#### 4.3.10.5 IBMDSwitcherClip::SetValid method

The **SetValid** method sets the clip name and frame count and sets the clip valid. **SetValid** has no effect unless all frames up to frameCount are valid. A valid clip can be downloaded and played by the media player.

##### Syntax

```
HRESULT          SetValid (string name, uint32_t frameCount);
```

##### Parameters

Name	Direction	Description
name	in	Clip name.
frameCount	in	Clip frame count.

##### Return Values

Value	Description
E_POINTER	The valid parameter is NULL.
S_OK	Success.

#### 4.3.10.6 IBMDSwitcherClip::SetInvalid method

The **SetInvalid** method invalidates every frame of a clip for all users, and should be done before uploading a new clip. A clip should then be set to valid once uploading is complete. This method will only be successful if you have a lock or no other connected client has a lock.

##### Syntax

```
HRESULT SetInvalid ();
```

##### Parameters

none.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 4.3.10.7 IBMDSwitcherClip::GetFrameCount method

The **GetFrameCount** method gets the current frame count of the clip.

##### Syntax

```
HRESULT      GetFrameCount (uint32_t* frameCount);
```

##### Parameters

Name	Direction	Description
name	out	Clip frame count.

##### Return Values

Value	Description
E_POINTER	The frameCount parameter is NULL.
S_OK	Success.

#### 4.3.10.8 IBMDSwitcherClip::GetMaxFrameCount method

The **GetMaxFrameCount** method gets the maximum frame count for the clip.

The maximum frame count can be set using **IBMDSwitcherMediaPool::SetClipMaxFrameCounts**.

##### Syntax

```
HRESULT GetMaxFrameCount (uint32_t* maxFrameCount);
```

##### Parameters

Name	Direction	Description
maxFrameCount	out	Maximum clip frame count.

##### Return Values

Value	Description
E_POINTER	The maxFrameCount parameter is NULL.
S_OK	Success.

#### 4.3.10.9 IBMDSwitcherClip::IsFrameValid method

The **IsFrameValid** method returns the validity of a clip frame.

##### Syntax

```
HRESULT      IsFrameValid (uint32_t frameIndex, bool* valid);
```

##### Parameters

Name	Direction	Description
frameIndex	in	Clip frame index.
valid	out	Validity of the clip frame.

##### Return Values

Value	Description
E_POINTER	The valid parameter is NULL.
E_INVALIDARG	The frameIndex parameter is invalid.
S_OK	Success.



#### 4.3.10.10 IBMDSwitcherClip::GetFrameHash method

The **GetFrameHash** method returns the hash of a clip frame.

##### Syntax

```
HRESULT          GetFrameHash (uint32_t frameIndex, BMDSwitcherHash* hash);
```

##### Parameters

Name	Direction	Description
frameIndex	in	Clip frame index.
hash	out	Clip frame hash.

##### Return Values

Value	Description
E_POINTER	The hash parameter is NULL.
E_INVALIDARG	The frameIndex parameter is invalid.
S_OK	Success.

#### 4.3.10.11 IBMDSwitcherClip::IsAudioValid method

The **IsAudioValid** method returns the validity of the clip audio. Valid clip audio can be downloaded and played by the media player.

##### Syntax

```
HRESULT          IsAudioValid (bool* valid);
```

##### Parameters

Name	Direction	Description
valid	out	Validity of the clip audio.

##### Return Values

Value	Description
E_POINTER	The valid parameter is NULL.
S_OK	Success.

#### 4.3.10.12 IBMDSwitcherClip::GetAudioName method

The **GetAudioName** method returns the name of the clip audio.

##### Syntax

```
HRESULT          GetAudioName (string* name);
```

##### Parameters

Name	Direction	Description
name	out	Clip audio name.

##### Return Values

Value	Description
E_POINTER	The name parameter is NULL.
S_OK	Success.

#### 4.3.10.13 IBMDSwitcherClip::SetAudioName method

The **SetAudioName** method sets the name of the clip audio.

##### Syntax

```
HRESULT          SetAudioName (string name);
```

##### Parameters

Name	Direction	Description
name	in	The still name.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The name parameter is invalid.

## 4.3.10.14 IBMDSwitcherClip::GetAudioHash method

The **GetAudioHash** method returns the hash of the clip audio.

### Syntax

```
HRESULT GetAudioHash (BMDSwitcherHash* hash);
```

### Parameters

Name	Direction	Description
hash	out	Clip audio frame hash.

### Return Values

Value	Description
E_POINTER	The hash parameter is NULL.
S_OK	Success.

## 4.3.10.15 IBMDSwitcherClip::SetAudioInvalid method

The **SetAudioInvalid** method invalidates the clip audio. Clip audio is set valid after a successful clip audio upload.

### Syntax

```
HRESULT SetAudioInvalid ();
```

### Parameters

none.

### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 4.3.10.16 IBMDSwitcherClip::Lock method

The **Lock** method obtains a client lock for the clip. Pass an object implementing the **IBMDSwitcherLockCallback** interface to receive **IBMDSwitcherLockCallback::Obtained** when the client obtains the clip lock.

##### Syntax

```
HRESULT Lock (IBMDSwitcherLockCallback* lockCallback);
```

##### Parameters

Name	Direction	Description
lockCallback	in	Callback object implementing the <b>IBMDSwitcherLockCallback</b> object interface.

##### Return Values

Value	Description
E_POINTER	The lockCallback parameter is invalid.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.10.17 IBMDSwitcherClip::Unlock method

The **Unlock** method releases the previous client lock for the clip.

##### Syntax

```
HRESULT          Unlock (IBMDSwitcherLockCallback* lockCallback);
```

##### Parameters

Name	Direction	Description
lockCallback	in	Callback object implementing the <b>IBMDSwitcherLockCallback</b> object interface.

##### Return Values

Value	Description
E_POINTER	The lockCallback parameter is invalid.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.10.18 IBMDSwitcherClip::UploadFrame method

The **UploadFrame** method transfers a clip frame to a clip in the media pool. The client must hold the clip lock for the duration of the transfer. No more than one transfer can occur at a time. Do not write to the frame's buffer during the transfer.

##### Syntax

```
HRESULT UploadFrame (uint32_t frameIndex, IBMDSwitcherFrame* frame);
```

##### Parameters

Name	Direction	Description
frameIndex	in	frameIndex
frame	in	Clip frame to upload. The frame dimensions must match the switcher video mode.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The frame parameter is invalid.
E_INVALIDARG	The frameIndex parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the clip lock.



#### 4.3.10.19 IBMDSwitcherClip::DownloadFrame method

The **DownloadFrame** method transfers a clip frame from the media pool. The client must hold the clip lock for the duration of the transfer. No more than one transfer can occur at a time.

##### Syntax

```
HRESULT      DownloadFrame (uint32_t frameIndex);
```

##### Parameters

Name	Direction	Description
frameIndex	in	Index of clip frame to download.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	The frameIndex parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the clip lock.

#### 4.3.10.20 IBMDSwitcherClip::UploadAudio method

The **UploadAudio** method transfers audio to a clip in the media pool. The client must hold the clip lock for the duration of the transfer. No more than one transfer can occur at a time. Do not write to the audio's buffer during the transfer.

##### Syntax

```
HRESULT UploadAudio (IBMDSwitcherAudio* audio);
```

##### Parameters

Name	Direction	Description
audio	in	Clip audio to upload.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The audio parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the clip lock.

#### 4.3.10.21 IBMDSwitcherClip::DownloadAudio method

The **Download** method transfers a clip from the media pool. The client must hold the clip lock for the duration of the transfer. No more than one transfer can occur at a time.

##### Syntax

```
HRESULT DownloadAudio ();
```

##### Parameters

none.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_OUTOFMEMORY	Unable to allocate required memory.
E_ACCESSDENIED	The client is not holding the clip lock.

#### 4.3.10.22 IBMDSwitcherClip::CancelTransfer method

The **CancelTransfer** method cancels the pending transfer. If there is no pending transfer then this method has no effect.

##### Syntax

```
HRESULT CancelTransfer ();
```

##### Parameters

none.

##### Return Values

Value	Description
S_OK	Success.

#### 4.3.10.23 IBMDSwitcherClip::GetProgress method

The **GetProgress** method gets the progress of the pending transfer. If there is no pending transfer then progress is zero.

##### Syntax

```
HRESULT GetProgress (double* progress);
```

##### Parameters

Name	Direction	Description
progress	out	Transfer progress. Range is between 0.0 and 1.0.

##### Return Values

Value	Description
E_POINTER	The progress parameter is invalid.
S_OK	Success.

#### 4.3.10.24 IBMDSwitcherClip::AddCallback method

The **AddCallback** method configures a callback to be called when an event occurs for an **IBMDSwitcherClip** object. Pass an object implementing the **IBMDSwitcherClipCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

**IBMDSwitcherClipCallback ::Notify** will be called immediately on the provided **IBMDSwitcherClipCallback** callback object with one of the following **BMDSwitcherMediaPoolEventType** eventTypes:

**bmdSwitcherMediaPoolEventTypeLockBusy**  
**bmdSwitcherMediaPoolEventTypeLockIdle**

#### Syntax

**HRESULT**           AddCallback (IBMDSwitcherClipCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherClipCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 4.3.10.25 IBMDSwitcherClip::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherClipCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherClipCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 4.3.11 IBMDSwitcherMediaPoolCallback Interface

The **IBMDSwitcherMediaPoolCallback** object interface is a callback class containing methods that are called when a property changes on an **IBMDSwitcherMediaPool** object. Like all callback methods, these callback methods may be called from another thread.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMediaPool	IID_IBMDSwitcherMediaPool	An <b>IBMDSwitcherMediaPoolCallback</b> object interface is installed with <b>IBMDSwitcherMediaPool::AddCallback</b> and removed with <b>IBMDSwitcherMediaPool::RemoveCallback</b>

Public Member Functions	
Method	Description
ClipFrameMaxCountsChanged	Called when the maximum frame count changes for one or more clips.
FrameTotalForClipsChanged	Called when the total number of frames available to clips changes.

## 4.3.11.1 IBMDSwitcherMediaPoolCallback::ClipFrameMaxCountsChanged method

The **ClipFrameMaxCountsChanged** method is called when the maximum frame count changes for one or more clips. Call **IBMDSwitcherMediaPool::GetClipMaxFrameCounts** to get the maximum frame counts for clips.

### Syntax

```
HRESULT          ClipFrameMaxCountsChanged ();
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.

## 4.3.11.2 IBMDSwitcherMediaPoolCallback::FrameTotalForClipsChanged method

The **FrameTotalForClipsChanged** method is called when the total number of frames available to clips changes. Call **IBMDSwitcherMediaPool::GetFrameTotalForClips** to get the the total number of frames available to clips.

### Syntax

```
HRESULT          FrameTotalForClipsChanged ();
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.



### 4.3.12 IBMDSwitcherMediaPool Interface

The **IBMDSwitcherMediaPool** object interface provides for the creation of frames and audio and for accessing and modifying stills and clips.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An <b>IBMDSwitcherMediaPool</b> object interface can be obtained with <b>IBMDSwitcher::QueryInterface</b> .

Public Member Functions	
Method	Description
GetStills	Gets the <b>IBMDSwitcherStills</b> object interface.
GetClip	Gets a <b>IBMDSwitcherClip</b> object interface.
GetClipCount	Gets the number of clips.
CreateFrame	Creates an <b>IBMDSwitcherFrame</b> object.
CreateAudio	Creates an <b>IBMDSwitcherAudio</b> object.
GetFrameTotalForClips	Gets the total number of frames available to clips.
GetClipMaxFrameCounts	Gets the maximum frame count for all clips.
SetClipMaxFrameCounts	Sets the maximum frame count for all clips.
AddCallback	Adds a media pool callback.
RemoveCallback	Removes a media pool callback.

#### 4.3.12.1 IBMDSwitcherMediaPool::GetStills method

The **GetStills** method gets the **IBMDSwitcherStills** object interface.

##### Syntax

```
HRESULT GetStills (IBMDSwitcherStills** stills);
```

##### Parameters

Name	Direction	Description
stills	out	The stills object interface.

##### Return Values

Value	Description
E_POINTER	The stills parameter is invalid.
S_OK	Success.

#### 4.3.12.2 IBMDSwitcherMediaPool::GetClip method

The **GetClip** method gets the **IBMDSwitcherClip** object interface.

##### Syntax

```
HRESULT GetClip (uint32_t clipIndex, IBMDSwitcherClip** clip);
```

##### Parameters

Name	Direction	Description
clipIndex	in	The clip index.
clip	out	The clip object interface.

##### Return Values

Value	Description
E_POINTER	The stills parameter is invalid.
E_INVALIDARG	The clipIndex parameter is invalid.
S_OK	Success.

# SECTION 4 Switcher Media

## 4.3.12.3 IBMDSwitcherMediaPool::GetClipCount method

The **GetClipCount** method gets the number of clips.

### Syntax

```
HRESULT GetClipCount (uint32_t* clipCount);
```

### Parameters

Name	Direction	Description
clipCount	out	The number of clips.

### Return Values

Value	Description
E_POINTER	The clipCount parameter is invalid.
S_OK	Success.

## 4.3.12.4 IBMDSwitcherMediaPool::CreateFrame method

The **CreateFrame** method creates an **IBMDSwitcherFrame** object.

### Syntax

```
HRESULT CreateFrame (BMDSwitcherPixelFormat pixelFormat, uint32_t width, uint32_t height,
                     IBMDSwitcherFrame** frame);
```

### Parameters

Name	Direction	Description
pixelFormat	in	The pixel format. See <b>BMDSwitcherPixelFormat</b> for a list of supported pixel formats.
width	in	The frame width in pixels.
height	in	The frame height in pixels.
frame	out	The newly created frame.

### Return Values

Value	Description
E_POINTER	The frame parameter is invalid.
E_INVALIDARG	The pixelFormat, width or height parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.12.5 IBMDSwitcherMediaPool::CreateAudio method

The **CreateAudio** method creates an **IBMDSwitcherAudio** object.

##### Syntax

```
HRESULT CreateAudio (uint32_t sizeBytes, IBMDSwitcherAudio** audio);
```

##### Parameters

Name	Direction	Description
sizeBytes	in	The audio's buffer size in bytes.
audio	out	The newly created audio object.

##### Return Values

Value	Description
E_POINTER	The audio parameter is invalid.
E_INVALIDARG	The sizeBytes parameter is invalid.
E_OUTOFMEMORY	Unable to allocate required memory.
E_FAIL	Failure.
S_OK	Success.

#### 4.3.12.6 IBMDSwitcherMediaPool::GetFrameTotalForClips method

The **GetFrameTotalForClips** method gets the total number of frames available to clips.

##### Syntax

```
HRESULT          GetFrameTotalForClips (uint32_t* total);
```

##### Parameters

Name	Direction	Description
clipCount	out	The number of clips.

##### Return Values

Value	Description
E_POINTER	The total parameter is invalid.
S_OK	Success.

#### 4.3.12.7 IBMDSwitcherMediaPool::GetClipMaxFrameCounts method

The **GetClipMaxFrameCounts** method gets the maximum frame count for all clips.

##### Syntax

```
HRESULT GetFrameTotalForClips (uint32_t clipCount, uint32_t* clipMaxFrameCounts);
```

##### Parameters

Name	Direction	Description
clipCount	in	Length of clipMaxFrameCounts array. This must match the switcher clip count.
clipMaxFrameCounts	out	A clipCount length array, where each element receives the maximum frame count for its respective clip index.

##### Return Values

Value	Description
E_POINTER	The clipMaxFrameCounts parameter is invalid.
E_INVALIDARG	The clipCount parameter is invalid.
S_OK	Success.



# SECTION 4 Switcher Media

## 4.3.12.8 IBMDSwitcherMediaPool::Clear method

The **Clear** method invalidates all stills, clips and clip audio.

### Syntax

```
HRESULT      Clear ();
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 4.3.12.9 IBMDSwitcherMediaPool::SetClipMaxFrameCounts method

The **SetClipMaxFrameCounts** method sets the maximum frame count for all clips.

##### Syntax

```
HRESULT SetClipMaxFrameCounts (uint32_t clipCount, const uint32_t* clipMaxFrameCounts);
```

##### Parameters

Name	Direction	Description
clipCount	in	Length of clipMaxFrameCounts array. This must match the switcher clip count.
clipMaxFrameCounts	in	A clipCount length array, where each element sets the maximum frame count for its respective clip index.

##### Return Values

Value	Description
E_POINTER	The clipMaxFrameCounts parameter is invalid.
E_INVALIDARG	The clipCount parameter is invalid.
S_OK	Success.

#### 4.3.12.10 IBMDSwitcherMediaPool::AddCallback method

The **AddCallback** method configures a callback to be called when an event occurs for an **IBMDSwitcherMediaPool** object. Pass an object implementing the **IBMDSwitcherMediaPoolCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT          SetClipMaxFrameCounts (uint32_t clipCount, const uint32_t* clipMaxFrameCounts);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMediaPoolCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 4.3.12.11 IBMDSwitcherMediaPool::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherMediaPoolCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMediaPoolCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## 5 Keyers

Any form of a keyer available in our switchers use these API components to perform chroma keying.

### 5.1 Key Data Types

#### 5.1.1 Key Type

**BMDSwitcherKeyType** enumerates the possible key types, used by **IBMDSwitcherKey** object interface.

<b>bmdSwitcherKeyTypeLuma</b>	Luminance key.
<b>bmdSwitcherKeyTypeChroma</b>	Chroma key.
<b>bmdSwitcherKeyTypePattern</b>	Pattern key.
<b>bmdSwitcherKeyTypeDVE</b>	DVE key.

#### 5.1.2 Fly Key Frames

**BMDSwitcherFlyKeyFrame** is a bit set that enumerates possible key frames for a fly key, used by **IBMDSwitcherKeyFlyParameters** object interface.

<b>bmdSwitcherFlyKeyFrameFull</b>	Full screen.
<b>bmdSwitcherFlyKeyFrameInfinityCentreOfKey</b>	Infinity at centre of key.
<b>bmdSwitcherFlyKeyFrameInfinityTopLeft</b>	Infinity at top left of screen.
<b>bmdSwitcherFlyKeyFrameInfinityTop</b>	Infinity at top centre of screen.
<b>bmdSwitcherFlyKeyFrameInfinityTopRight</b>	Infinity at top right of screen.
<b>bmdSwitcherFlyKeyFrameInfinityLeft</b>	Infinity at mid left of screen.
<b>bmdSwitcherFlyKeyFrameInfinityCentre</b>	Infinity at centre of screen.
<b>bmdSwitcherFlyKeyFrameInfinityRight</b>	Infinity at mid right of screen.
<b>bmdSwitcherFlyKeyFrameInfinityBottomLeft</b>	Infinity at bottom left of screen.
<b>bmdSwitcherFlyKeyFrameInfinityBottom</b>	Infinity at bottom centre of screen.
<b>bmdSwitcherFlyKeyFrameInfinityBottomRight</b>	Infinity at bottom right of screen.
<b>bmdSwitcherFlyKeyFrameA</b>	User-defined key frame A.
<b>bmdSwitcherFlyKeyFrameB</b>	User-defined key frame B.

### 5.1.3 Border Bevel Options

**BMDSwitcherBorderBevelOption** enumerates possible border bevel style options.

This type is used by **IBMDSwitcherKeyDVEParameters** and **IBMDSwitcherInputSuperSource** object interfaces.

<b>bmdSwitcherBorderBevelOptionNone</b>	No bevel.
<b>bmdSwitcherBorderBevelOptionInOut</b>	Both inner and outer bevel.
<b>bmdSwitcherBorderBevelOptionIn</b>	Inner bevel only.
<b>bmdSwitcherBorderBevelOptionOut</b>	Outer bevel only.

### 5.1.4 Key Event Type

**BMDSwitcherKeyEventType** enumerates the possible event types for **IBMDSwitcherKeyCallback**.

<b>bmdSwitcherKeyEventTypeTypeChanged</b>	The type changed.
<b>bmdSwitcherKeyEventTypeInputCutChanged</b>	The cut input source changed.
<b>bmdSwitcherKeyEventTypeInputFillChanged</b>	The fill input source changed.
<b>bmdSwitcherKeyEventTypeOnAirChanged</b>	The on-air flag changed.
<b>bmdSwitcherKeyEventTypeCanBeDVEKeyChanged</b>	The can-be-DVE flag changed.
<b>bmdSwitcherKeyEventTypeMaskedChanged</b>	The masked flag changed.
<b>bmdSwitcherKeyEventTypeMaskTopChanged</b>	The mask top value changed.
<b>bmdSwitcherKeyEventTypeMaskBottomChanged</b>	The mask bottom value changed.
<b>bmdSwitcherKeyEventTypeMaskLeftChanged</b>	The mask left value changed.
<b>bmdSwitcherKeyEventTypeMaskRightChanged</b>	The mask right value changed.

### 5.1.5 Luminance Key Parameters Event Type

**BMDSwitcherKeyLumaParametersEventType** enumerates the possible event types for **IBMDSwitcherKeyLumaParametersCallback**.

<b>bmdSwitcherKeyLumaParametersEventTypePreMultipliedChanged</b>	The pre-multiplied flag changed.
<b>bmdSwitcherKeyLumaParametersEventTypeClipChanged</b>	The clip value changed.
<b>bmdSwitcherKeyLumaParametersEventTypeGainChanged</b>	The gain value changed.
<b>bmdSwitcherKeyLumaParametersEventTypeInverseChanged</b>	The inverse flag changed.

### 5.1.6 Chroma Key Parameters Event Type

**BMDSwitcherKeyChromaParametersEventType** enumerates the possible event types for **IBMDSwitcherKeyChromaParametersCallback**.

<b>bmdSwitcherKeyChromaParametersEventTypeHueChanged</b>	The hue value changed.
<b>bmdSwitcherKeyChromaParametersEventTypeGainChanged</b>	The gain value changed.
<b>bmdSwitcherKeyChromaParametersEventTypeYSuppressChanged</b>	The y-suppress value changed.
<b>bmdSwitcherKeyChromaParametersEventTypeLiftChanged</b>	The lift value changed.
<b>bmdSwitcherKeyChromaParametersEventTypeNarrowChanged</b>	The narrow flag changed.

### 5.1.7 Pattern Key Parameters Event Type

**BMDSwitcherKeyPatternParametersEventType** enumerates the possible event types for **IBMDSwitcherKeyPatternParametersCallback**.

<b>bmdSwitcherKeyPatternParametersEventTypePatternChanged</b>	The pattern style changed.
<b>bmdSwitcherKeyPatternParametersEventTypeSizeChanged</b>	The size value changed.
<b>bmdSwitcherKeyPatternParametersEventTypeSymmetryChanged</b>	The symmetry value changed.
<b>bmdSwitcherKeyPatternParametersEventTypeSoftnessChanged</b>	The softness value changed.
<b>bmdSwitcherKeyPatternParametersEventTypeHorizontalOffsetChanged</b>	The horizontal offset changed.
<b>bmdSwitcherKeyPatternParametersEventTypeVerticalOffsetChanged</b>	The vertical offset changed.
<b>bmdSwitcherKeyPatternParametersEventTypeInverseChanged</b>	The inverse flag changed.

## 5.1.8 DVE Key Parameters Event Type

**BMDSwitcherKeyDVEParametersEventType** enumerates the possible event types for **IBMDSwitcherKeyDVEParametersCallback**.

**bmdSwitcherKeyDVEParametersEventTypeShadowChanged**

The shadow flag changed.

**bmdSwitcherKeyDVEParametersEventTypeLightSourceDirectionChanged**

The light source direction value changed.

**bmdSwitcherKeyDVEParametersEventTypeLightSourceAltitudeChanged**

The light source altitude value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderEnabledChanged**

The border enabled flag changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderBevelChanged**

The border bevel option changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderWidthInChanged**

The border inner width value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderWidthOutChanged**

The border outer width value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderSoftnessInChanged**

The border inner softness value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderSoftnessOutChanged**

The border outer softness value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderBevelSoftnessChanged**

The border bevel softness value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderBevelPositionChanged**

The border bevel position value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderOpacityChanged**

The border opacity value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderHueChanged**

The border hue value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderSaturationChanged**

The border saturation value changed.

**bmdSwitcherKeyDVEParametersEventTypeBorderLumaChanged**

The border luminance value changed.

**bmdSwitcherKeyDVEParametersEventTypeMaskedChanged**

The masked flag changed.

**bmdSwitcherKeyDVEParametersEventTypeMaskTopChanged**

The mask top value changed.

**bmdSwitcherKeyDVEParametersEventTypeMaskBottomChanged**

The mask bottom value changed.

**bmdSwitcherKeyDVEParametersEventTypeMaskLeftChanged**

The mask left value changed.

**bmdSwitcherKeyDVEParametersEventTypeMaskRightChanged**

The mask right value changed.



### 5.1.9 Fly Key Parameters Event Type

**BMDSwitcherKeyFlyParametersEventType** enumerates the possible event types for **IBMDSwitcherKeyFlyParametersCallback**.

<b>bmdSwitcherKeyFlyParametersEventTypeFlyChanged</b>	The fly flag changed.
<b>bmdSwitcherKeyFlyParametersEventTypeCanFlyChanged</b>	The can-fly flag changed.
<b>bmdSwitcherKeyFlyParametersEventTypeRateChanged</b>	The rate value changed.
<b>bmdSwitcherKeyFlyParametersEventTypeSizeXChanged</b>	The size x value changed.
<b>bmdSwitcherKeyFlyParametersEventTypeSizeYChanged</b>	The size y value changed.
<b>bmdSwitcherKeyFlyParametersEventTypePositionXChanged</b>	The position x value changed.
<b>bmdSwitcherKeyFlyParametersEventTypePositionYChanged</b>	The position y value changed.
<b>bmdSwitcherKeyFlyParametersEventTypeRotationChanged</b>	The rotation value changed.
<b>bmdSwitcherKeyFlyParametersEventTypelsKeyFrameStoredChanged</b>	The is-key-frame-stored flag changed.
<b>bmdSwitcherKeyFlyParametersEventTypelsAtKeyFramesChanged</b>	The is-at-key-frames status changed.
<b>bmdSwitcherKeyFlyParametersEventTypelsRunningChanged</b>	The is-running status changed.

#### 5.1.10 Downstream Key Event Type

**BMDSwitcherDownstreamKeyEventType** enumerates the possible event types for **IBMDSwitcherDownstreamKeyCallback**.

<b>bmdSwitcherDownstreamKeyEventTypeInputCutChanged</b>	The cut input source changed.
<b>bmdSwitcherDownstreamKeyEventTypeInputFillChanged</b>	The fill input source changed.
<b>bmdSwitcherDownstreamKeyEventTypeTieChanged</b>	The tie flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeRateChanged</b>	The rate value changed.
<b>bmdSwitcherDownstreamKeyEventTypeOnAirChanged</b>	The on-air flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeIsTransitioningChanged</b>	The is-transitioning flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeIsAutoTransitioningChanged</b>	The is-auto-transitioning flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeFramesRemainingChanged</b>	The frames remaining value changed.
<b>bmdSwitcherDownstreamKeyEventTypePreMultipliedChanged</b>	The pre-multiplied flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeClipChanged</b>	The clip value changed.
<b>bmdSwitcherDownstreamKeyEventTypeGainChanged</b>	The gain value changed.
<b>bmdSwitcherDownstreamKeyEventTypeInverseChanged</b>	The inverse flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeMaskedChanged</b>	The masked flag changed.
<b>bmdSwitcherDownstreamKeyEventTypeMaskTopChanged</b>	The mask top value changed.
<b>bmdSwitcherDownstreamKeyEventTypeMaskBottomChanged</b>	The mask bottom value changed.
<b>bmdSwitcherDownstreamKeyEventTypeMaskLeftChanged</b>	The mask left value changed.
<b>bmdSwitcherDownstreamKeyEventTypeMaskRightChanged</b>	The mask right value changed.

5.2

Interface Reference

5.2.1

IBMDSwitcherKeylterator Interface

The **IBMDSwitcherKeylterator** is used to enumerate the available keys for each mix effect block.

A reference to an **IBMDSwitcherKeylterator** object interface may be obtained from an **IBMDSwitcherMixEffectBlock** object interface using the **Createlterator** method. Pass **IID\_IBMDSwitcherKeylterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	<b>IBMDSwitcherMixEffectBlock::Createlterator</b> can return an <b>IBMDSwitcherKeylterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherKey object interface.

### 5.2.1.1 IBMDSwitcherMediaPool::AddCallback method

The **Next** method returns the next available **IBMDSwitcherKey** object interface.

#### Syntax

```
HRESULT         Next (IBMDSwitcherKey** key);
```

#### Parameters

Name	Direction	Description
key	out	<b>IBMDSwitcherKey</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherKey</b> objects available.
E_POINTER	The key parameter is invalid.

### 5.2.2 IBMDSwitcherKey Interface

The **IBMDSwitcherKey** object interface is used for manipulating the basic settings of a key. Please note that the mask settings in this interface only apply to luminance, chroma and pattern key types; DVE type key uses its own mask settings available in the **IBMDSwitcherKeyDVEParameters** interface.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyIterator	IID_IBMDSwitcherKeyIterator	An <b>IBMDSwitcherKey</b> object will be returned after a successful call to <b>IBMDSwitcherKeyIterator::Next</b> method.

Public Member Functions	
Method	Description
GetType	Get the current key type.
SetType	Set the key type.
GetInputCut	Get the current cut input source.
SetInputCut	Set the cut input source.
GetInputFill	Get the current fill input source.
SetInputFill	Set the fill input source.
GetFillInputAvailabilityMask	Get the availability mask for the fill of this input.
GetCutInputAvailabilityMask	Get the availability mask for the cut of this input.
GetOnAir	Get the on-air flag.
SetOnAir	Set the on-air flag.
GetCanBeDVEKey	Determine if this key can be set to DVE type.
GetMasked	Get the current masked flag.
SetMasked	Set the masked flag.

Public Member Functions	
Method	Description
GetMaskTop	Get the current mask top value.
SetMaskTop	Set the mask top value.
GetMaskBottom	Get the current mask bottom value.
SetMaskBottom	Set the mask bottom value.
GetMaskLeft	Get the current mask left value.
SetMaskLeft	Set the mask left value.
GetMaskRight	Get the current mask right value.
SetMaskRight	Set the mask right value.
ResetMask	Reset mask to default values.
GetTransitionSelectionMask	Get the corresponding <b>BMDSwitcherTransitionSelection</b> bit mask for this key.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

### 5.2.2.1 IBMDSwitcherKey::GetType method

The **GetType** method returns the current key type.

#### Syntax

```
HRESULT      GetType (BMDSwitcherKeyType* type);
```

#### Parameters

Name	Direction	Description
type	out	The current key type.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The type parameter is invalid.
E_UNEXPECTED	Unexpected error occurred.

### 5.2.2.2 IBMDSwitcherKey::SetType method

The **SetType** method sets the key to the specified type.

#### Syntax

```
HRESULT      SetType (BMDSwitcherKeyType type);
```

#### Parameters

Name	Direction	Description
type	in	The desired key type.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The type parameter is invalid.



### 5.2.2.3 IBMDSwitcherKey::GetInputCut method

The **GetInputCut** method returns the selected cut input source.

#### Syntax

```
HRESULT      GetInputCut (BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
inputId	out	<b>BMDSwitcherInputId</b> of the selected cut input source.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inputId parameter is invalid.

#### 5.2.2.4 IBMDSwitcherKey::SetInputCut method

The **SetInputCut** method sets the cut input source.

##### Syntax

```
HRESULT      SetInputCut (BMDSwitcherInputId inputId);
```

##### Parameters

Name	Direction	Description
inputId	in	The desired cut input source's <b>BMDSwitcherInputId</b> .

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The inputId parameter is invalid.

### 5.2.2.5 IBMDSwitcherKey::GetInputFill method

The **GetInputFill** method returns the selected fill input source.

#### Syntax

```
HRESULT      GetInputFill (BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
inputId	out	<b>BMDSwitcherInputId</b> of the selected fill input source.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inputId parameter is invalid.

### 5.2.2.6 **IBMDSwitcherKey::SetInputFill** method

The **SetInputFill** method sets the fill input source.

#### Syntax

**HRESULT**            **SetInputFill** (BMDSwitcherInputId inputId);

#### Parameters

Name	Direction	Description
inputId	in	The desired fill input source's <b>BMDSwitcherInputId</b> .

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The inputId parameter is invalid.

### 5.2.2.7 IBMDSwitcherKey::GetFillInputAvailabilityMask method

The GetFillInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for fill inputs available to this key. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a fill input for this key.

#### Syntax

**HRESULT**            GetFillInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

## 5.2.2.8 IBMDSwitcherKey::GetCutInputAvailabilityMask method

The GetCutInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for cut inputs available to this key. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a cut input for this key.

### Syntax

**HRESULT** GetCutInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

### 5.2.2.9 IBMDSwitcherKey::GetOnAir method

The **GetOnAir** method returns the on-air flag.

#### Syntax

```
HRESULT GetOnAir (boolean* onAir);
```

#### Parameters

Name	Direction	Description
onAir	out	Boolean on-air flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The onAir parameter is invalid.

### 5.2.2.10 IBMDSwitcherKey::SetOnAir method

The **SetOnAir** method sets the on-air flag.

#### Syntax

```
HRESULT SetOnAir (boolean onAir);
```

#### Parameters

Name	Direction	Description
onAir	in	The desired on-air flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.2.11 IBMDSwitcherKey::CanBeDVEKey method

The **CanBeDVEKey** method returns a status flag of whether this key can be set to the DVE type. The DVE hardware is a shared resource; if another component is currently using the resource, it may not be available for this key.

#### Syntax

```
HRESULT CanBeDVEKey (boolean* canDVE);
```

#### Parameters

Name	Direction	Description
canDVE	out	Boolean status of whether this key can be a DVE key.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The canDVE parameter is invalid.



## 5.2.2.12 IBMDSwitcherKey::GetMasked method

The **GetMasked** method returns whether masking is enabled or not.

### Syntax

```
HRESULT GetMasked (boolean* masked);
```

### Parameters

Name	Direction	Description
masked	out	Boolean flag of whether masking is enabled.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The masked parameter is invalid.

## 5.2.2.13 IBMDSwitcherKey::SetMasked method

Use **SetMasked** method to enable or disable masking.

### Syntax

```
HRESULT SetMasked (boolean masked);
```

### Parameters

Name	Direction	Description
masked	in	The desired masked value.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.2.14 **IBMDSwitcherKey::GetMaskTop method**

The **GetMaskTop** method returns the current mask top value.

##### **Syntax**

```
HRESULT GetMaskTop (double* maskTop);
```

##### **Parameters**

Name	Direction	Description
maskTop	out	The current mask top value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskTop parameter is invalid.

#### 5.2.2.15 **IBMDSwitcherKey::SetMaskTop method**

The **SetMaskTop** method sets the mask top value.

##### **Syntax**

```
HRESULT SetMaskTop (double maskTop);
```

##### **Parameters**

Name	Direction	Description
maskTop	in	The desired mask top value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.2.16 **IBMDSwitcherKey::GetMaskBottom** method

The **GetMaskBottom** method returns the current mask bottom value.

##### **Syntax**

```
HRESULT GetMaskBottom (double* maskBottom);
```

##### **Parameters**

Name	Direction	Description
maskBottom	out	The current mask bottom value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskBottom parameter is invalid.

#### 5.2.2.17 **IBMDSwitcherKey::SetMaskBottom** method

The **SetMaskBottom** method sets the mask bottom value.

##### **Syntax**

```
HRESULT SetMaskBottom (double maskBottom);
```

##### **Parameters**

Name	Direction	Description
maskBottom	in	The desired mask bottom value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.2.18 **IBMDSwitcherKey::GetMaskLeft** method

The **GetMaskLeft** method returns the current mask left value.

##### **Syntax**

```
HRESULT GetMaskLeft (double* maskLeft);
```

##### **Parameters**

Name	Direction	Description
maskLeft	out	The current mask left value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskLeft parameter is invalid.

#### 5.2.2.19 **IBMDSwitcherKey::SetMaskLeft** method

The **SetMaskLeft** method sets the mask left value.

##### **Syntax**

```
HRESULT SetMaskLeft (double maskLeft);
```

##### **Parameters**

Name	Direction	Description
maskLeft	in	The desired mask left value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.2.20 **IBMDSwitcherKey::GetMaskRight** method

The **GetMaskRight** method returns the current mask right value.

##### **Syntax**

```
HRESULT GetMaskRight (double* maskRight);
```

##### **Parameters**

Name	Direction	Description
maskRight	out	The current mask right value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskRight parameter is invalid.

#### 5.2.2.21 **IBMDSwitcherKey::SetMaskRight** method

The **SetMaskRight** method sets the mask right value.

##### **Syntax**

```
HRESULT SetMaskRight (double maskRight);
```

##### **Parameters**

Name	Direction	Description
maskRight	in	The desired mask right value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 5.2.2.22 IBMDSwitcherKey::ResetMask method

Use the **ResetMask** method to reset mask settings to default values.

### Syntax

```
HRESULT      ResetMask (void);
```

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 5.2.2.23 IBMDSwitcherKey::GetTransitionSelectionMask method

The **GetTransitionSelectionMask** method returns the corresponding **BMDSwitcherTransitionSelection** bit mask for this key.

### Syntax

```
HRESULT      GetTransitionSelectionMask (BMDSwitcherTransitionSelection* selectionMask);
```

### Parameters

Name	Direction	Description
selectionMask	out	<b>BMDSwitcherTransitionSelection</b> bit mask.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The selectionMask parameter is invalid.

#### 5.2.2.24 IBMDSwitcherKey::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKey** object. Pass an object implementing the **IBMDSwitcherKeyCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT AddCallback (IBMDSwitcherKeyCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 5.2.2.25 IBMDSwitcherKey::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



5.2.3

IBMDSwitcherKeyCallback Interface

The **IBMDSwitcherKeyCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKey** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyCallback</b> object interface is installed with <b>IBMDSwitcherKey::AddCallback</b> and removed with <b>IBMDSwitcherKey::RemoveCallback</b> .

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.3.1 IBMDSwitcherKeyCallback::Notify method

The **Notify** method is called when **IBMDSwitcherKey** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherKeyEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.4 IBMDSwitcherKeyLumaParameters Interface

The **IBMDSwitcherKeyLumaParameters** object interface is used for manipulating parameters specific to luminance type key.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyLumaParameters</b> object interface can be obtained with <b>IBMDSwitcherKey::QueryInterface</b> .

Public Member Functions	
Method	Description
GetPreMultiplied	Get the current pre-multiplied flag.
SetPreMultiplied	Set pre-multiplied flag.
GetClip	Get the current clip value.
SetClip	Set the clip value.
GetGain	Get the current gain value.
SetGain	Set gain value.
GetInverse	Get the current inverse flag.
SetInverse	Set the inverse flag.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

### 5.2.4.1 IBMDSwitcherKeyLumaParameters::GetPreMultiplied method

The **GetPreMultiplied** method returns the current pre-multiplied flag.

#### Syntax

```
HRESULT      GetPreMultiplied (boolean* preMultiplied);
```

#### Parameters

Name	Direction	Description
preMultiplied	out	The current pre-multiplied flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The preMultiplied parameter is invalid.

#### 5.2.4.2 **IBMDSwitcherKeyLumaParameters::SetPreMultiplied** method

The **SetPreMultiplied** method sets the pre-multiplied flag.

Note that clip, gain and inverse controls are not used when pre-multiplied flag is set to true.

##### **Syntax**

```
HRESULT          SetPreMultiplied (boolean preMultiplied);
```

##### **Parameters**

Name	Direction	Description
preMultiplied	in	The desired pre-multiplied flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.4.3 **IBMDSwitcherKeyLumaParameters::GetClip method**

The **GetClip** method returns the current clip value.

##### **Syntax**

```
HRESULT GetClip (double* clip);
```

##### **Parameters**

Name	Direction	Description
clip	out	The current clip value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The clip parameter is invalid.

#### 5.2.4.4 **IBMDSwitcherKeyLumaParameters::SetClip method**

The **SetClip** method sets the clip value.

##### **Syntax**

```
HRESULT SetClip (double clip);
```

##### **Parameters**

Name	Direction	Description
clip	in	The desired clip value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.4.5 **IBMDSwitcherKeyLumaParameters::GetGain method**

The **GetGain** method returns the current gain value.

##### **Syntax**

```
HRESULT GetGain (double* gain);
```

##### **Parameters**

Name	Direction	Description
gain	out	The current gain value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

#### 5.2.4.6 **IBMDSwitcherKeyLumaParameters::SetGain method**

The **SetGain** method sets the gain value.

##### **Syntax**

```
HRESULT SetGain (double gain);
```

##### **Parameters**

Name	Direction	Description
gain	in	The desired gain value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.4.7 **IBMDSwitcherKeyLumaParameters::GetInverse method**

The **GetInverse** method returns the current inverse flag.

##### **Syntax**

**HRESULT**            **GetInverse** (boolean\* inverse);

##### **Parameters**

Name	Direction	Description
inverse	out	The current inverse flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

#### 5.2.4.8 **IBMDSwitcherKeyLumaParameters::SetInverse method**

The **SetInverse** method sets the inverse flag.

##### **Syntax**

**HRESULT**            **SetInverse** (boolean inverse);

##### **Parameters**

Name	Direction	Description
inverse	in	The desired inverse flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.4.9 IBMDSwitcherKeyLumaParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyLumaParameters** object. Pass an object implementing the **IBMDSwitcherKeyLumaParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT          AddCallback (IBMDSwitcherKeyLumaParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyLumaParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 5.2.4.10 IBMDSwitcherKeyLumaParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyLumaParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyLumaParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.5 IBMDSwitcherKeyLumaParametersCallback Interface

The **IBMDSwitcherKeyLumaParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyLumaParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyLumaParameters	IID_IBMDSwitcherKeyLumaParameters	An <b>IBMDSwitcherKeyLumaParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyLumaParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyLumaParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.5.1 **IBMDSwitcherKeyLumaParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherKeyLumaParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

```
HRESULT Notify (BMDSwitcherKeyLumaParametersEventType eventType);
```

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyLumaParametersEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.6 IBMDSwitcherKeyChromaParameters Interface

The **IBMDSwitcherKeyChromaParameters** object interface is used for manipulating settings specific to the chroma type key.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyChromaParameters</b> object interface can be obtained with <b>IBMDSwitcherKey::QueryInterface</b> .

Public Member Functions	
Method	Description
GetHue	Get the current hue value.
SetHue	Set the hue value.
GetGain	Get the current gain value.
SetGain	Set the gain value.
GetYSuppress	Get the current y-suppress flag.
SetYSuppress	Set the y-suppress flag.
GetLift	Get the current lift value.
SetLift	Set the lift value.
GetNarrow	Get the current narrow flag.
SetNarrow	Set the narrow flag.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

### 5.2.6.1 **IBMDSwitcherKeyChromaParameters::GetHue method**

The **GetHue** method gets the current hue value.

#### **Syntax**

```
HRESULT GetHue (double* hue);
```

#### **Parameters**

Name	Direction	Description
hue	out	The current hue value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The hue parameter is invalid.

### 5.2.6.2 **IBMDSwitcherKeyChromaParameters::SetHue method**

The **SetHue** method sets the hue value.

#### **Syntax**

```
HRESULT SetHue (double hue);
```

#### **Parameters**

Name	Direction	Description
hue	in	The desired hue value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.6.3 **IBMDSwitcherKeyChromaParameters::GetGain method**

The **GetGain** method gets the current gain value.

#### **Syntax**

```
HRESULT GetGain (double* gain);
```

#### **Parameters**

Name	Direction	Description
gain	out	The current gain value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

### 5.2.6.4 **IBMDSwitcherKeyChromaParameters::SetGain method**

The **SetGain** method sets the gain value.

#### **Syntax**

```
HRESULT SetGain (double gain);
```

#### **Parameters**

Name	Direction	Description
gain	in	The desired gain value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.6.5 **IBMDSwitcherKeyChromaParameters::GetYSuppress** method

The **GetYSuppress** method gets the current y-suppress value.

##### **Syntax**

```
HRESULT          GetYSuppress (double* ySuppress);
```

##### **Parameters**

Name	Direction	Description
ySuppress	out	The current y-suppress value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The ySuppress parameter is invalid.

#### 5.2.6.6 **IBMDSwitcherKeyChromaParameters::SetYSuppress** method

The **SetYSuppress** method sets the y-suppress value.

##### **Syntax**

```
HRESULT          SetYSuppress (double ySuppress);
```

##### **Parameters**

Name	Direction	Description
ySuppress	in	The desired ySuppress value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 5.2.6.7 **IBMDSwitcherKeyChromaParameters::GetLift** method

The **GetLift** method gets the current lift value.

#### **Syntax**

```
HRESULT GetLift (double* lift);
```

#### **Parameters**

Name	Direction	Description
lift	out	The current lift value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The lift parameter is invalid.

### 5.2.6.8 **IBMDSwitcherKeyChromaParameters::SetLift** method

The **SetLift** method sets the lift value.

#### **Syntax**

```
HRESULT SetLift (double lift);
```

#### **Parameters**

Name	Direction	Description
lift	in	The desired lift value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.6.9 **IBMDSwitcherKeyChromaParameters::GetNarrow** method

The **GetNarrow** method gets the current narrow flag.

#### Syntax

```
HRESULT GetNarrow (boolean* narrow);
```

#### Parameters

Name	Direction	Description
narrow	out	The current narrow flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The narrow parameter is invalid.

### 5.2.6.10 **IBMDSwitcherKeyChromaParameters::SetNarrow** method

The **SetNarrow** method sets the narrow flag.

#### Syntax

```
HRESULT SetNarrow (boolean narrow);
```

#### Parameters

Name	Direction	Description
narrow	in	The desired narrow flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.6.11 IBMDSwitcherKeyChromaParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyChromaParameters** object. Pass an object implementing the **IBMDSwitcherKeyChromaParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT AddCallback (IBMDSwitcherKeyChromaParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyChromaParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 5.2.6.12 **IBMDSwitcherKeyChromaParameters::RemoveCallback** method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyChromaParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyChromaParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.7 IBMDSwitcherKeyChromaParametersCallback Interface

The **IBMDSwitcherKeyChromaParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyChromaParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyChromaParameters	IID_IBMDSwitcherKeyChromaParameters	An <b>IBMDSwitcherKeyChromaParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyChromaParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyChromaParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.7.1 **IBMDSwitcherKeyChromaParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherKeyChromaParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherKeyChromaParametersEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyChromaParametersEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.8 IBMDSwitcherKeyPatternParameters Interface

The **IBMDSwitcherKeyPatternParameters** object interface is used for manipulating settings specific to the pattern type key.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyPatternParameters</b> object interface can be obtained with <b>IBMDSwitcherKey::QueryInterface</b> .

Public Member Functions	
Method	Description
GetPattern	Get the current pattern style.
SetPattern	Set the pattern style.
GetSize	Get the current size value.
SetSize	Set the size value.
GetSymmetry	Get the current symmetry value.
SetSymmetry	Set the symmetry value.
GetSoftness	Get the current softness value.
SetSoftness	Set the softness value.
GetHorizontalOffset	Get the current horizontal offset.
SetHorizontalOffset	Set the horizontal offset.
GetVerticalOffset	Get the current vertical offset.
SetVerticalOffset	Set the vertical offset.
GetInverse	Get the current inverse flag.
SetInverse	Set the inverse flag.

# SECTION 5 Keyers

Public Member Functions	
Method	Description
AddCallback	Add a callback.
RemoveCallback	Remove a callback.



### 5.2.8.1 IBMDSwitcherKeyPatternParameters::GetPattern method

The **GetPattern** method gets the current pattern style.

#### Syntax

```
HRESULT      GetPattern (BMDSwitcherPatternStyle* pattern);
```

#### Parameters

Name	Direction	Description
pattern	out	The current pattern style of <b>BMDSwitcherPatternStyle</b> .

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The pattern parameter is invalid.

### 5.2.8.2 IBMDSwitcherKeyPatternParameters::SetPattern method

The **SetPattern** method sets the pattern style.

#### Syntax

```
HRESULT          SetPattern (BMDSwitcherPatternStyle pattern);
```

#### Parameters

Name	Direction	Description
pattern	in	The desired <b>BMDSwitcherPatternStyle</b> pattern style.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The pattern parameter is invalid.
E_FAIL	Failure.

### 5.2.8.3 **IBMDSwitcherKeyPatternParameters::GetSize method**

The **GetSize** method gets the current size value.

#### **Syntax**

```
HRESULT          GetSize (double* size);
```

#### **Parameters**

Name	Direction	Description
size	out	The current size value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The size parameter is invalid.

### 5.2.8.4 **IBMDSwitcherKeyPatternParameters::SetSize method**

The **SetSize** method sets the size value.

#### **Syntax**

```
HRESULT          SetSize (double size);
```

#### **Parameters**

Name	Direction	Description
size	in	The desired size value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.8.5 **IBMDSwitcherKeyPatternParameters::GetSymmetry** method

The **GetSymmetry** method gets the current symmetry value.

##### **Syntax**

```
HRESULT GetSymmetry (double* symmetry);
```

##### **Parameters**

Name	Direction	Description
symmetry	out	The current symmetry value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The symmetry parameter is invalid.

#### 5.2.8.6 **IBMDSwitcherKeyPatternParameters::SetSymmetry** method

The **SetSymmetry** method sets the symmetry value.

##### **Syntax**

```
HRESULT SetSymmetry (double symmetry);
```

##### **Parameters**

Name	Direction	Description
symmetry	in	The desired symmetry value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.8.7 **IBMDSwitcherKeyPatternParameters::GetSoftness** method

The **GetSoftness** method gets the current softness value.

##### **Syntax**

```
HRESULT GetSoftness (double* softness);
```

##### **Parameters**

Name	Direction	Description
softness	out	The current softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The softness parameter is invalid.

#### 5.2.8.8 **IBMDSwitcherKeyPatternParameters::SetSoftness** method

The **SetSoftness** method sets the softness value.

##### **Syntax**

```
HRESULT SetSoftness (double softness);
```

##### **Parameters**

Name	Direction	Description
softness	in	The desired softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.8.9 **IBMDSwitcherKeyPatternParameters::GetHorizontalOffset** method

The **GetHorizontalOffset** method gets the current horizontal offset value.

##### **Syntax**

```
HRESULT GetHorizontalOffset (double* hOffset);
```

##### **Parameters**

Name	Direction	Description
hOffset	out	The current horizontal offset value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The hOffset parameter is invalid.

#### 5.2.8.10 **IBMDSwitcherKeyPatternParameters::SetHorizontalOffset** method

The **SetHorizontalOffset** method sets the horizontal offset value.

##### **Syntax**

```
HRESULT SetHorizontalOffset (double hOffset);
```

##### **Parameters**

Name	Direction	Description
hOffset	in	The desired horizontal offset value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.8.11 **IBMDSwitcherKeyPatternParameters::GetVerticalOffset** method

The **GetVerticalOffset** method gets the current vertical offset value.

##### **Syntax**

```
HRESULT GetVerticalOffset (double* vOffset);
```

##### **Parameters**

Name	Direction	Description
vOffset	out	The current vertical offset value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The vOffset parameter is invalid.

#### 5.2.8.12 **IBMDSwitcherKeyPatternParameters::SetVerticalOffset** method

The **SetVerticalOffset** method sets the vertical offset value.

##### **Syntax**

```
HRESULT SetVerticalOffset (double vOffset);
```

##### **Parameters**

Name	Direction	Description
vOffset	in	The desired vertical offset value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.8.13 **IBMDSwitcherKeyPatternParameters::GetInverse** method

The **GetInverse** method gets the current inverse flag.

##### **Syntax**

```
HRESULT GetInverse (boolean* inverse);
```

##### **Parameters**

Name	Direction	Description
inverse	out	The current inverse flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

#### 5.2.8.14 **IBMDSwitcherKeyPatternParameters::SetInverse** method

The **SetInverse** method sets the inverse flag.

##### **Syntax**

```
HRESULT SetInverse (boolean inverse);
```

##### **Parameters**

Name	Direction	Description
inverse	in	The desired inverse flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.8.15 IBMDSwitcherKeyPatternParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyPatternParameters** object. Pass an object implementing the **IBMDSwitcherKeyPatternParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT      AddCallback (IBMDSwitcherKeyPatternParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyPatternParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 5.2.8.16 IBMDSwitcherKeyPatternParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyPatternParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyPatternParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.9

IBMDSwitcherKeyPatternParametersCallback Interface

The **IBMDSwitcherKeyPatternParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyPatternParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyPatternParameters	IID_IBMDSwitcherKeyPatternParameters	An <b>IBMDSwitcherKeyPatternParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyPatternParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyPatternParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.9.1 **IBMDSwitcherKeyPatternParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherKeyPatternParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherKeyPatternParametersEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyPatternParametersEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.10 IBMDSwitcherKeyDVEParameters Interface

The **IBMDSwitcherKeyDVEParameters** object interface is used for manipulating settings specific to the DVE-type key. Note that properties that affect a fly key also affects a DVE key; they are access through the **IBMDSwitcherKeyFlyParameters** object interface. Also note that the mask properties in this interface only affect keys with their type set to DVE.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyDVEParameters</b> object interface can be obtained with <b>IBMDSwitcherKey::QueryInterface</b> .

Public Member Functions	
Method	Description
GetShadow	Get the current shadow flag.
SetShadow	Set the shadow flag.
GetLightSourceDirection	Get the current light source direction value.
SetLightSourceDirection	Set the light source direction value.
GetLightSourceAltitude	Get the current light source altitude value.
SetLightSourceAltitude	Set the light source altitude value.
GetBorderEnabled	Get the current border enabled flag.
SetBorderEnabled	Set the border enabled flag.
GetBorderBevel	Get the current border bevel option.
SetBorderBevel	Set the border bevel option.
GetBorderWidthIn	Get the current border inner width value.
SetBorderWidthIn	Set the border inner width value.
GetBorderWidthOut	Get the current border outer width value.

Public Member Functions	
Method	Description
SetBorderWidthOut	Set the border outer width value.
GetBorderSoftnessIn	Get the current border inner softness value.
SetBorderSoftnessIn	Set the border inner softness value.
GetBorderSoftnessOut	Get the current border outer softness value.
SetBorderSoftnessOut	Set the border outer softness value.
GetBorderBevelSoftness	Get the current border bevel softness value.
SetBorderBevelSoftness	Set the border bevel softness value.
GetBorderBevelPosition	Get the current border bevel position value.
SetBorderBevelPosition	Set the border bevel position value.
GetBorderOpacity	Get the current border opacity value.
SetBorderOpacity	Set the border opacity value.
GetBorderHue	Get the current border hue value.
SetBorderHue	Set the border hue value.
GetBorderSaturation	Get the current border saturation value.
SetBorderSaturation	Set the border saturation value.
GetBorderLuma	Get the current border luminance value.
SetBorderLuma	Set the border luminance value.
GetMasked	Get the current masked flag.
SetMasked	Set the masked flag.
GetMaskTop	Get the current mask top value.
SetMaskTop	Set the mask top value.
GetMaskBottom	Get the current mask bottom value.

Public Member Functions	
Method	Description
SetMaskBottom	Set the mask bottom value.
GetMaskLeft	Get the current mask left value.
SetMaskLeft	Set the mask left value.
GetMaskRight	Get the current mask right value.
SetMaskRight	Set the mask right value.
ResetMask	Reset mask properties to default values.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

#### 5.2.10.1 **IBMDSwitcherKeyDVEParameters::GetShadow method**

The **GetShadow** method gets the current shadow flag.

##### **Syntax**

```
HRESULT          GetShadow (boolean* shadow);
```

##### **Parameters**

Name	Direction	Description
shadow	out	The current shadow flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The shadow parameter is invalid.

#### 5.2.10.2 **IBMDSwitcherKeyDVEParameters::SetShadow method**

The **SetShadow** method sets the shadow flag.

##### **Syntax**

```
HRESULT          SetShadow (boolean shadow);
```

##### **Parameters**

Name	Direction	Description
shadow	in	The desired shadow flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 5.2.10.3 **IBMDSwitcherKeyDVEParameters::GetLightSourceDirection** method

The **GetLightSourceDirection** method gets the current light source direction value.

#### **Syntax**

**HRESULT** GetLightSourceDirection (double\* degrees);

#### **Parameters**

Name	Direction	Description
degrees	out	The current light source direction in degrees.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The degrees parameter is invalid.

### 5.2.10.4 **IBMDSwitcherKeyDVEParameters::SetLightSourceDirection** method

The **SetLightSourceDirection** method sets the light source direction value.

#### **Syntax**

**HRESULT** SetLightSourceDirection (double degrees);

#### **Parameters**

Name	Direction	Description
degrees	in	The desired light source direction value in degrees.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.5 **IBMDSwitcherKeyDVEParameters::GetLightSourceAltitude** method

The **GetLightSourceAltitude** method gets the current light source altitude value.

##### **Syntax**

```
HRESULT GetLightSourceAltitude (double* altitude);
```

##### **Parameters**

Name	Direction	Description
altitude	out	The current light source altitude value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The altitude parameter is invalid.

#### 5.2.10.6 **IBMDSwitcherKeyDVEParameters::SetLightSourceAltitude** method

The **SetLightSourceAltitude** method sets the light source altitude value.

##### **Syntax**

```
HRESULT SetLightSourceAltitude (double altitude);
```

##### **Parameters**

Name	Direction	Description
altitude	in	The desired light source altitude value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.7 **IBMDSwitcherKeyDVEParameters::GetBorderEnabled** method

The **GetBorderEnabled** method gets the current border enabled flag.

##### **Syntax**

**HRESULT** GetBorderEnabled (boolean\* enabled);

##### **Parameters**

Name	Direction	Description
enabled	out	The current border enabled flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The enabled parameter is invalid.

#### 5.2.10.8 **IBMDSwitcherKeyDVEParameters::SetBorderEnabled** method

The **SetBorderEnabled** method sets the border enabled flag.

##### **Syntax**

**HRESULT** SetBorderEnabled (boolean enabled);

##### **Parameters**

Name	Direction	Description
enabled	in	The desired border enabled flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.9 **IBMDSwitcherKeyDVEParameters::GetBorderBevel** method

The **GetBorderBevel** method gets the current border bevel option.

#### Syntax

```
HRESULT          GetBorderBevel (BMDSwitcherBorderBevelOption* bevelOption);
```

#### Parameters

Name	Direction	Description
bevelOption	out	The current bevel option of <b>BMDSwitcherBorderBevelOption</b> .

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The bevelOption parameter is invalid.
E_UNEXPECTED	Unexpected error occurred.

### 5.2.10.10 **IBMDSwitcherKeyDVEParameters::SetBorderBevel** method

The **SetBorderBevel** method sets the border bevel option.

#### Syntax

```
HRESULT          SetBorderBevel (BMDSwitcherBorderBevelOption bevelOption);
```

#### Parameters

Name	Direction	Description
bevelOption	in	The desired bevel option of <b>BMDSwitcherBorderBevelOption</b> .

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The bevelOption parameter is invalid.
E_FAIL	Failure.

### 5.2.10.11 **IBMDSwitcherKeyDVEParameters::GetBorderWidthIn** method

The **GetBorderWidthIn** method gets the current border inner width value.

#### Syntax

```
HRESULT GetBorderWidthIn (double* widthIn);
```

#### Parameters

Name	Direction	Description
widthIn	out	The current border inner width value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The widthIn parameter is invalid.

### 5.2.10.12 **IBMDSwitcherKeyDVEParameters::SetBorderWidthIn** method

The **SetBorderWidthIn** method sets the border inner width value.

#### Syntax

```
HRESULT SetBorderWidthIn (double widthIn);
```

#### Parameters

Name	Direction	Description
widthIn	in	The desired border inner width value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.13 **IBMDSwitcherKeyDVEParameters::GetBorderWidthOut** method

The **GetBorderWidthOut** method gets the current border outer width value.

##### **Syntax**

**HRESULT** GetBorderWidthOut (double\* widthOut);

##### **Parameters**

Name	Direction	Description
widthIn	out	The current border outer width value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The widthOut parameter is invalid.

#### 5.2.10.14 **IBMDSwitcherKeyDVEParameters::SetBorderWidthOut** method

The **SetBorderWidthOut** method sets the border outer width value.

##### **Syntax**

**HRESULT** SetBorderWidthOut (double widthOut);

##### **Parameters**

Name	Direction	Description
widthIn	in	The desired border outer width value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.15 **IBMDSwitcherKeyDVEParameters::GetBorderSoftnessIn** method

The **GetBorderSoftnessIn** method gets the current border inner softness value.

##### **Syntax**

```
HRESULT GetBorderSoftnessIn (double* softIn);
```

##### **Parameters**

Name	Direction	Description
softIn	out	The current border inner softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The softIn parameter is invalid.

#### 5.2.10.16 **IBMDSwitcherKeyDVEParameters::SetBorderSoftnessIn** method

The **SetBorderSoftnessIn** method sets the border inner softness value.

##### **Syntax**

```
HRESULT SetBorderSoftnessIn (double softIn);
```

##### **Parameters**

Name	Direction	Description
softIn	in	The desired border inner softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.10.17 **IBMDSwitcherKeyDVEParameters::GetBorderSoftnessOut** method

The **GetBorderSoftnessOut** method gets the current border outer softness value.

##### **Syntax**

**HRESULT**            GetBorderSoftnessOut (double\* softOut);

##### **Parameters**

Name	Direction	Description
softOut	out	The current border outer softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The softOut parameter is invalid.

#### 5.2.10.18 **IBMDSwitcherKeyDVEParameters::SetBorderSoftnessOut** method

The **SetBorderSoftnessOut** method sets the border outer softness value.

##### **Syntax**

**HRESULT**            SetBorderSoftnessOut (double softOut);

##### **Parameters**

Name	Direction	Description
softOut	in	The desired border outer softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.19 **IBMDSwitcherKeyDVEParameters::GetBorderBevelSoftness** method

The **GetBorderBevelSoftness** method gets the current border bevel softness value.

#### Syntax

```
HRESULT GetBorderBevelSoftness (double* bevelSoft);
```

#### Parameters

Name	Direction	Description
bevelSoft	out	The current border bevel softness value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The bevelSoft parameter is invalid.

### 5.2.10.20 **IBMDSwitcherKeyDVEParameters::SetBorderBevelSoftness** method

The **SetBorderBevelSoftness** method sets the border bevel softness value.

#### Syntax

```
HRESULT SetBorderBevelSoftness (double bevelSoft);
```

#### Parameters

Name	Direction	Description
bevelSoft	in	The desired border bevel softness value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.21 **IBMDSwitcherKeyDVEParameters::GetBorderBevelPosition** method

The **GetBorderBevelPosition** method gets the current border bevel position value.

#### **Syntax**

```
HRESULT GetBorderBevelPosition (double* bevelPosition);
```

#### **Parameters**

Name	Direction	Description
bevelPosition	out	The current border bevel position value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The bevelPosition parameter is invalid.

### 5.2.10.22 **IBMDSwitcherKeyDVEParameters::SetBorderBevelPosition** method

The **SetBorderBevelPosition** method sets the border bevel position value.

#### **Syntax**

```
HRESULT SetBorderBevelPosition (double bevelPosition);
```

#### **Parameters**

Name	Direction	Description
bevelPosition	in	The desired border bevel position value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.23 **IBMDSwitcherKeyDVEParameters::GetBorderOpacity method**

The **GetBorderOpacity** method gets the current border opacity value.

##### **Syntax**

```
HRESULT GetBorderOpacity (double* opacity);
```

##### **Parameters**

Name	Direction	Description
opacity	out	The current border opacity value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The opacity parameter is invalid.

#### 5.2.10.24 **IBMDSwitcherKeyDVEParameters::SetBorderOpacity method**

The **SetBorderOpacity** method sets the border opacity value.

##### **Syntax**

```
HRESULT SetBorderOpacity (double opacity);
```

##### **Parameters**

Name	Direction	Description
opacity	in	The desired border opacity value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.25 **IBMDSwitcherKeyDVEParameters::GetBorderHue** method

The **GetBorderHue** method gets the current border hue value.

##### **Syntax**

```
HRESULT GetBorderHue (double* hue);
```

##### **Parameters**

Name	Direction	Description
hue	out	The current border hue value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The hue parameter is invalid.

#### 5.2.10.26 **IBMDSwitcherKeyDVEParameters::SetBorderHue** method

The **SetBorderHue** method sets the border hue value.

##### **Syntax**

```
HRESULT SetBorderHue (double hue);
```

##### **Parameters**

Name	Direction	Description
hue	in	The desired border hue value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.27 **IBMDSwitcherKeyDVEParameters::GetBorderSaturation** method

The **GetBorderSaturation** method gets the current border saturation value.

#### Syntax

```
HRESULT GetBorderSaturation (double* sat);
```

#### Parameters

Name	Direction	Description
sat	out	The current border saturation value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The sat parameter is invalid.

### 5.2.10.28 **IBMDSwitcherKeyDVEParameters::SetBorderSaturation** method

The **SetBorderSaturation** method sets the border saturation value.

#### Syntax

```
HRESULT SetBorderSaturation (double saturation);
```

#### Parameters

Name	Direction	Description
saturation	in	The desired border saturation value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.29 **IBMDSwitcherKeyDVEParameters::GetBorderLuma** method

The **GetBorderLuma** method gets the current border luminance value.

#### **Syntax**

```
HRESULT GetBorderLuma (double* luma);
```

#### **Parameters**

Name	Direction	Description
luma	out	The current border luminance value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The luma parameter is invalid.

### 5.2.10.30 **IBMDSwitcherKeyDVEParameters::SetBorderLuma** method

The **SetBorderLuma** method sets the border luminance value.

#### **Syntax**

```
HRESULT SetBorderLuma (double luma);
```

#### **Parameters**

Name	Direction	Description
luma	in	The desired border luminance value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.31 **IBMDSwitcherKeyDVEParameters::GetMasked** method

The **GetMasked** method returns whether masking is enabled or not.

#### Syntax

```
HRESULT GetMasked (boolean* masked);
```

#### Parameters

Name	Direction	Description
masked	out	Boolean flag of whether masking is enabled.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The masked parameter is invalid.

### 5.2.10.32 **IBMDSwitcherKeyDVEParameters::SetMasked** method

Use **SetMasked** method to enable or disable masking.

#### Syntax

```
HRESULT SetMasked (boolean masked);
```

#### Parameters

Name	Direction	Description
masked	in	The desired masked value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 5.2.10.33 **IBMDSwitcherKeyDVEParameters::GetMaskTop** method

The **GetMaskTop** method returns the current mask top value.

#### **Syntax**

```
HRESULT GetMaskTop (double* maskTop);
```

#### **Parameters**

Name	Direction	Description
maskTop	out	The current mask top value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskTop parameter is invalid.

### 5.2.10.34 **IBMDSwitcherKeyDVEParameters::SetMaskTop** method

The **SetMaskTop** method sets the mask top value.

#### **Syntax**

```
HRESULT SetMaskTop (double maskTop);
```

#### **Parameters**

Name	Direction	Description
maskTop	in	The desired mask top value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.35 **IBMDSwitcherKeyDVEParameters::GetMaskBottom** method

The **GetMaskBottom** method returns the current mask bottom value.

#### Syntax

```
HRESULT GetMaskBottom (double* maskBottom);
```

#### Parameters

Name	Direction	Description
maskBottom	out	The current mask bottom value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskBottom parameter is invalid.

### 5.2.10.36 **IBMDSwitcherKeyDVEParameters::SetMaskBottom** method

The **SetMaskBottom** method sets the mask bottom value.

#### Syntax

```
HRESULT SetMaskBottom (double maskBottom);
```

#### Parameters

Name	Direction	Description
maskBottom	in	The desired mask bottom value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.37 **IBMDSwitcherKeyDVEParameters::GetMaskLeft** method

The **GetMaskLeft** method returns the current mask left value.

#### Syntax

```
HRESULT GetMaskLeft (double* maskLeft);
```

#### Parameters

Name	Direction	Description
maskLeft	out	The current mask left value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskLeft parameter is invalid.

### 5.2.10.38 **BMDSwitcherKeyDVEParameters::SetMaskLeft** method

The **SetMaskLeft** method sets the mask left value.

#### Syntax

```
HRESULT SetMaskLeft (double maskLeft);
```

#### Parameters

Name	Direction	Description
maskLeft	in	The desired mask left value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.10.39 **BMDSwitcherKeyDVEParameters::GetMaskRight** method

The **GetMaskRight** method returns the current mask right value.

#### Syntax

```
HRESULT GetMaskRight (double* maskRight);
```

#### Parameters

Name	Direction	Description
maskRight	out	The current mask right value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskRight parameter is invalid.

### 5.2.10.40 **BMDSwitcherKeyDVEParameters::SetMaskRight** method

The **SetMaskRight** method sets the mask right value.

#### Syntax

```
HRESULT SetMaskRight (double maskRight);
```

#### Parameters

Name	Direction	Description
maskRight	in	The desired mask right value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.41 IBMDSwitcherKeyDVEParameters::ResetMask method

The **ResetMask** method resets the mask settings to default values.

##### Syntax

```
HRESULT      ResetMask (void);
```

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.10.42 IBMDSwitcherKeyDVEParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyDVEParameters** object. Pass an object implementing the **IBMDSwitcherKeyDVEParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

**HRESULT**           AddCallback (IBMDSwitcherKeyDVEParametersCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyDVEParametersCallback</b> object interface.

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 5.2.10.43 IBMDSwitcherKeyDVEParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyDVEParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyDVEParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.11

IBMDSwitcherKeyDVEParametersCallback Interface

The **IBMDSwitcherKeyDVEParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyDVEParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyDVEParameters	IID_IBMDSwitcherKeyDVEParameters	An <b>IBMDSwitcherKeyDVEParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyDVEParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyDVEParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.



### 5.2.11.1 **IBMDSwitcherKeyDVEParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherKeyDVEParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherKeyDVEParametersEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyDVEParametersEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12 IBMDSwitcherKeyFlyParameters Interface

The **IBMDSwitcherKeyFlyParameters** object interface is used for manipulating fly settings of a key. A luminance, chroma or pattern key can be made a “fly” key, filtering its current state through the DVE hardware. Turning off the fly setting will remove the filter and return the key to its original state. Note that most properties in this interface also take effect when the key type is set to DVE.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <b>IBMDSwitcherKeyFlyParameters</b> object interface can be obtained with <b>IBMDSwitcherKey::QueryInterface</b> .

Public Member Functions	
Method	Description
GetFly	Get the current fly flag.
SetFly	Set the fly flag.
GetCanFly	Get the current can-fly flag.
GetRate	Get the current fly rate.
SetRate	Set the fly rate.
GetSizeX	Get the current size x value.
SetSizeX	Set the size x value.
GetSizeY	Get the current size y value.
SetSizeY	Set the size y value.
GetPositionX	Get the current position x value.
SetPositionX	Set the position x value.
GetPositionY	Get the current position y value.
SetPositionY	Set the position y value.

# SECTION 5 Keyers

Public Member Functions	
Method	Description
GetRotation	Get the current rotation value.
SetRotation	Set the rotation value.
ResetRotation	Reset rotation to default value.
ResetDVE	Reset DVE properties (size, position and rotation) to default values.
ResetDVEFull	Reset DVE properties (size, position and rotation) to full screen.
IsKeyFrameStored	Determine if a key frame has been stored.
StoreAsKeyFrame	Store current state into a key frame.
RunToKeyFrame	Run to a key frame.
IsAtKeyFrames	Determines if the current frame matches any of the stored key frames.
IsRunning	Determines if the key is currently running.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

## 5.2.12.1 IBMDSwitcherKeyFlyParameters::GetFly method

The **GetFly** method returns whether fly is enabled or not.

### Syntax

```
HRESULT      GetFly (boolean* isFlyKey);
```

### Parameters

Name	Direction	Description
isFlyKey	out	Boolean status of whether fly is enabled.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The isFlyKey parameter is invalid.

## 5.2.12.2 IBMDSwitcherKeyFlyParameters::SetFly method

Use the **SetFly** method to enable or disable fly.

### Syntax

```
HRESULT      SetFly (boolean isFlyKey);
```

### Parameters

Name	Direction	Description
isFlyKey	in	The desired fly enable flag.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.3 IBMDSwitcherKeyFlyParameters::GetCanFly method

The **GetCanFly** method returns whether this key can enable fly or not. The DVE hardware is a shared resource; if another component is currently using the resource, it may not be available for this key.

#### Syntax

```
HRESULT GetCanFly (boolean* canFly);
```

#### Parameters

Name	Direction	Description
canFly	out	Boolean status of the can-fly flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The canFly parameter is invalid.

#### 5.2.12.4 IBMDSwitcherKeyFlyParameters::GetRate method

The **GetRate** method gets the current fly rate value.

##### Syntax

```
HRESULT          GetRate (uint32_t* frames);
```

##### Parameters

Name	Direction	Description
frames	out	The current rate value in frames.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

### 5.2.12.5 IBMDSwitcherKeyFlyParameters::SetRate method

The **SetRate** method sets the fly rate value.

#### Syntax

```
HRESULT      SetRate (uint32_t frames);
```

#### Parameters

Name	Direction	Description
frames	in	The desired rate value in frames.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The frames parameter is invalid.
E_FAIL	Failure.

### 5.2.12.6 IBMDSwitcherKeyFlyParameters::GetSizeX method

The **GetSizeX** method gets the current size x value. The flying size is a multiple of the original key size.

#### Syntax

```
HRESULT      GetSizeX (double* multiplierX);
```

#### Parameters

Name	Direction	Description
multiplierX	out	The current size x value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The multiplierX parameter is invalid.



### 5.2.12.7 IBMDSwitcherKeyFlyParameters::SetSizeX method

The **SetSizeX** method sets the size x value. The flying size is a multiple of the original key size.

#### Syntax

```
HRESULT      SetSizeX (double multiplierX);
```

#### Parameters

Name	Direction	Description
multiplierX	in	The desired size x value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.12.8 **IBMDSwitcherKeyFlyParameters::GetSizeY method**

The **GetSizeY** method gets the current size y value. The flying size is a multiple of the original key size.

##### **Syntax**

```
HRESULT          GetSizeY (double* multiplierY);
```

##### **Parameters**

Name	Direction	Description
multiplierY	out	The current size y value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The multiplierY parameter is invalid.

### 5.2.12.9 IBMDSwitcherKeyFlyParameters::SetSizeY method

The **SetSizeY** method sets the size y value. The flying size is a multiple of the original key size.

#### Syntax

```
HRESULT      SetSizeY (double multiplierY);
```

#### Parameters

Name	Direction	Description
multiplierY	in	The desired size y value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.10 IBMDSwitcherKeyFlyParameters::GetPositionX method

The **GetPositionX** method gets the current position x value. This is an offset from the original key position.

#### Syntax

```
HRESULT      GetPositionX (double* offsetX);
```

#### Parameters

Name	Direction	Description
offsetX	out	The current offset x value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The offsetX parameter is invalid.

### 5.2.12.11 IBMDSwitcherKeyFlyParameters::SetPositionX method

The **SetPositionX** method sets the position x value. This is an offset from the original key position.

#### Syntax

```
HRESULT      SetPositionX (double offsetX);
```

#### Parameters

Name	Direction	Description
offsetX	in	The desired position x value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.12 **IBMDSwitcherKeyFlyParameters::GetPositionY** method

The **GetPositionY** method gets the current position y value. This is an offset from the original key position.

#### **Syntax**

```
HRESULT GetPositionY (double* offsetY);
```

#### **Parameters**

Name	Direction	Description
offsetY	out	The current offset y value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The offsetY parameter is invalid.

#### 5.2.12.13 **IBMDSwitcherKeyFlyParameters::SetPositionY method**

The **SetPositionY** method sets the position y value. This is an offset from the original key position.

##### **Syntax**

**HRESULT**            **SetPositionY** (double offsetY);

##### **Parameters**

Name	Direction	Description
offsetY	in	The desired position y value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.12.14 **IBMDSwitcherKeyFlyParameters::GetRotation method**

The **GetRotation** method gets the current rotation value.

##### **Syntax**

**HRESULT**            **GetRotation** (double\* degrees);

##### **Parameters**

Name	Direction	Description
degrees	out	The current rotation value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The degrees parameter is invalid.

### 5.2.12.15 **IBMDSwitcherKeyFlyParameters::SetRotation** method

The **SetRotation** method sets the rotation value.

#### **Syntax**

```
HRESULT SetRotation (double degrees);
```

#### **Parameters**

Name	Direction	Description
degrees	in	The desired rotation value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.16 **IBMDSwitcherKeyFlyParameters::ResetRotation** method

The **ResetRotation** method resets the rotation value to its default.

#### **Syntax**

```
HRESULT ResetRotation (void);
```

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 5.2.12.17 **IBMDSwitcherKeyFlyParameters::ResetDVE method**

The **ResetDVE** method resets the DVE parameters to their default values, i.e. size, position and rotation.

#### **Syntax**

```
HRESULT          ResetDVE (void);
```

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.18 **IBMDSwitcherKeyFlyParameters::ResetDVEFull method**

The **ResetDVEFull** method resets the key fly parameters to full screen with no rotation.

#### **Syntax**

```
HRESULT          ResetDVEFull (void);
```

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.12.19 IBMDSwitcherKeyFlyParameters::IsKeyFrameStored method

The **IsKeyFrameStored** method returns whether the specified key frame has been stored or not. It is intended for use with user-defined key frames to determine if they have been stored.

#### Syntax

```
HRESULT IsKeyFrameStored (BMDSwitcherFlyKeyFrame keyFrame, boolean* stored);
```

#### Parameters

Name	Direction	Description
keyFrame	in	Specify a single key frame of <b>BMDSwitcherFlyKeyFrame</b> to query the status on.
stored	out	The current status flag of whether the specified key frame has been stored.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The keyFrame parameter is invalid.
E_POINTER	The stored parameter is invalid.

### 5.2.12.20 **IBMDSwitcherKeyFlyParameters::StoreAsKeyFrame** method

The **StoreAsKeyFrame** method stores the current frame into the specified key frame(s). Multiple user-defined key frames can be specified.

#### Syntax

```
HRESULT StoreAsKeyFrame (BMDSwitcherFlyKeyFrame keyFrames);
```

#### Parameters

Name	Direction	Description
keyFrames	in	Specify where to store the current frame, must be user-defined key frame(s).

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The keyFrames parameter is invalid.
E_FAIL	Failure.

### 5.2.12.21 IBMDSwitcherKeyFlyParameters::RunToKeyFrame method

The **RunToKeyFrame** method commences a run from current frame to the specified key frame.

#### Syntax

```
HRESULT          RunToKeyFrame (BMDSwitcherFlyKeyFrame destination);
```

#### Parameters

Name	Direction	Description
destination	in	The destination key frame.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The destination parameter is invalid.
E_FAIL	Failure.

### 5.2.12.22 **IBMDSwitcherKeyFlyParameters::IsAtKeyFrames** method

The **IsAtKeyFrames** method returns a bit set of key frames that match the current frame. Zero is returned if the current frame does not match any built-in or user-defined frames.

#### **Syntax**

```
HRESULT IsAtKeyFrames (BMDSwitcherFlyKeyFrame* keyFrames);
```

#### **Parameters**

Name	Direction	Description
keyFrames	out	All key frames that match the current frame.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The keyFrames parameter is invalid.

### 5.2.12.23 IBMDSwitcherKeyFlyParameters::GetKeyFrameParameters method

The **GetKeyFrameParameters** method returns an object interface for accessing individual parameters in a key frame.

#### Syntax

```
HRESULT      GetKeyFrameParameters (BMDSwitcherFlyKeyFrame keyFrame, IBMDSwitcherKeyFlyKeyFrameParameters**
                                     keyFrameParameters);
```

#### Parameters

Name	Direction	Description
keyFrame	in	The desired key frame.
keyFrameParameters	out	<b>IBMDSwitcherKeyFlyKeyFrameParameters</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The keyFrame parameter is invalid.
E_POINTER	The keyFrameParameters parameter is invalid.

#### 5.2.12.24 IBMDSwitcherKeyFlyParameters::IsRunning method

The **IsRunning** method returns the current run status.

##### Syntax

```
HRESULT      IsRunning (boolean* isRunning, BMDSwitcherFlyKeyFrame* destination);
```

##### Parameters

Name	Direction	Description
isRunning	out	Boolean status of whether the key is running.
destination	out	If the key is running, this is the destination of the run.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The isRunning and/or destination parameter is invalid.
E_FAIL	Failure.

### 5.2.12.25 IBMDSwitcherKeyFlyParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyFlyParameters** object. Pass an object implementing the **IBMDSwitcherKeyFlyParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherKeyFlyParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyFlyParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



### 5.2.12.26 IBMDSwitcherKeyFlyParameters::RemoveCallback method

The RemoveCallback method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyFlyParametersCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyFlyParametersCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.13 IBMDSwitcherKeyFlyParametersCallback Interface

The **IBMDSwitcherKeyFlyParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyFlyParameters** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyFlyParameters	IID_IBMDSwitcherKeyFlyParameters	An <b>IBMDSwitcherKeyFlyParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyFlyParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyFlyParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.13.1 **IBMDSwitcherKeyFlyParametersCallback::Notify method**

The **Notify** method is called when **IBMDSwitcherKeyFlyParameters** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

**HRESULT** Notify (BMDSwitcherKeyFlyParametersEventType eventType, BMDSwitcherFlyKeyFrame keyFrame);

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyFlyParametersEventType</b> that describes the type of event that has occurred.
keyFrame	in	This parameter is only valid when eventType is <b>bmdSwitcherKeyFlyParametersEventTypes KeyFrameStoredChanged</b> , it specifies the changed key frame.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.14 IBMDSwitcherKeyFlyKeyFrameParameters Interface

The IBMDSwitcherKeyFlyKeyFrameParameters object interface provides access to individual key frame parameters.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyFlyParameters	IID_IBMDSwitcherKeyFlyParameters	An <b>IBMDSwitcherKeyFlyKeyFrameParameters</b> object interface can be obtained from <b>IBMDSwitcherKeyFlyParameters::GetKeyFrameParameters</b> .

Public Member Functions	
Method	Description
GetSizeX	Get the size x value.
SetSizeX	Set the size x value.
GetSizeY	Get the size y value.
SetSizeY	Set the size y value.
GetPositionX	Get the position x value.
SetPositionX	Set the position x value.
GetPositionY	Get the position y value.
SetPositionY	Set the position y value.
GetRotation	Get the rotation value.
SetRotation	Set the rotation value.
GetBorderWidthOut	Get the border outer width value.
SetBorderWidthOut	Set the border outer width value.
GetBorderWidthIn	Get the border inner width value.
SetBorderWidthIn	Set the border inner width value.
GetBorderSoftnessOut	Get the border outer softness value.

Public Member Functions	
Method	Description
SetBorderSoftnessOut	Set the border outer softness value.
GetBorderSoftnessIn	Get the border inner softness value.
SetBorderSoftnessIn	Set the border inner softness value.
GetBorderBevelSoftness	Get the border bevel softness value.
SetBorderBevelSoftness	Set the border bevel softness value.
GetBorderBevelPosition	Get the border bevel position value.
SetBorderBevelPosition	Set the border bevel position value.
GetBorderOpacity	Get the border opacity value.
SetBorderOpacity	Set the border opacity value.
GetBorderHue	Get the border hue value.
SetBorderHue	Set the border hue value.
GetBorderSaturation	Get the border saturation value.
SetBorderSaturation	Set the border saturation value.
GetBorderLuma	Get the border luminance value.
SetBorderLuma	Set the border luminance value.
GetBorderLightSourceDirection	Get the border light source direction value.
SetBorderLightSourceDirection	Set the border light source direction value.
GetBorderLightSourceAltitude	Get the border light source altitude value.
SetBorderLightSourceAltitude	Set the border light source altitude value.
GetMaskTop	Get the mask top value.
SetMaskTop	Set the mask top value.
GetMaskBottom	Get the mask bottom value.
SetMaskBottom	Set the mask bottom value.

Public Member Functions	
Method	Description
GetMaskLeft	Get the mask left value.
SetMaskLeft	Set the mask left value.
GetMaskRight	Get the mask right value.
SetMaskRight	Set the mask right value.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

#### 5.2.14.1 **IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeX** method

The **GetSizeX** method gets the size x value.

##### **Syntax**

```
HRESULT          GetSizeX (double* multiplierX);
```

##### **Parameters**

Name	Direction	Description
multiplierX	out	The current size x value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The multiplierX parameter is invalid.

#### 5.2.14.2 **IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeX** method

The **SetSizeX** method sets the size x value.

##### **Syntax**

```
HRESULT          SetSizeX (double multiplierX);
```

##### **Parameters**

Name	Direction	Description
multiplierX	in	The desired size x value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.3 **IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeY** method

The **GetSizeY** method gets the size y value.

##### **Syntax**

```
HRESULT          GetSizeY (double* multiplierY);
```

##### **Parameters**

Name	Direction	Description
multiplierY	out	The size y value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The multiplierY parameter is invalid.

#### 5.2.14.4 **IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeY** method

The **SetSizeY** method sets the size y value.

##### **Syntax**

```
HRESULT          SetSizeY (double multiplierY);
```

##### **Parameters**

Name	Direction	Description
multiplierY	in	The desired size y value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.14.5 IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionX method

The **GetPositionX** method gets the position x value.

##### Syntax

```
HRESULT      GetPositionX (double* offsetX);
```

##### Parameters

Name	Direction	Description
offsetX	out	The position x value.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The offsetX parameter is invalid.

#### 5.2.14.6 IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionX method

The **SetPositionX** method sets the position x value.

##### Syntax

```
HRESULT      SetPositionX (double offsetX);
```

##### Parameters

Name	Direction	Description
offsetX	in	The desired position x value.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.14.7 **IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionY** method

The **GetPositionY** method gets the position y value.

**Syntax**

```
HRESULT      GetPositionY (double* offsetY);
```

**Parameters**

Name	Direction	Description
offsetY	out	The position y value.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The offsetY parameter is invalid.

5.2.14.8 **IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionY** method

The **SetPositionY** method sets the position y value.

**Syntax**

```
HRESULT      SetPositionY (double offsetY);
```

**Parameters**

Name	Direction	Description
offsetY	in	The desired position y value.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.9 IBMDSwitcherKeyFlyKeyFrameParameters::GetRotation method

The **GetRotation** method gets the rotation value.

##### Syntax

**HRESULT** GetRotation (double\* degrees);

##### Parameters

Name	Direction	Description
degrees	out	The rotation value.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The degrees parameter is invalid.

#### 5.2.14.10 IBMDSwitcherKeyFlyKeyFrameParameters::SetRotation method

The **SetRotation** method sets the rotation value.

##### Syntax

**HRESULT** SetRotation (double degrees);

##### Parameters

Name	Direction	Description
degrees	in	The desired rotation value.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.11 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthOut** method

The **GetBorderWidthOut** method gets the border outer width value.

##### **Syntax**

**HRESULT**            GetBorderWidthOut (double\* widthOut);

##### **Parameters**

Name	Direction	Description
widthOut	out	The border outer width value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The widthOut parameter is invalid.

#### 5.2.14.12 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthOut** method

The **SetBorderWidthOut** method sets the border outer width value.

##### **Syntax**

**HRESULT**            SetBorderWidthOut (double widthOut);

##### **Parameters**

Name	Direction	Description
widthOut	in	The desired border outer width value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.14.13 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthIn method**

The **GetBorderWidthIn** method gets the border inner width value.

**Syntax**

```
HRESULT      GetBorderWidthIn (double* widthIn);
```

**Parameters**

Name	Direction	Description
widthIn	out	The border inner width value.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The widthIn parameter is invalid.

5.2.14.14 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthIn method**

The **SetBorderWidthIn** method sets the border inner width value.

**Syntax**

```
HRESULT      SetBorderWidthIn (double widthIn);
```

**Parameters**

Name	Direction	Description
widthIn	in	The desired border inner width value.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.15 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessOut** method

The **GetBorderSoftnessOut** method gets the border outer softness value.

##### **Syntax**

**HRESULT**            GetBorderSoftnessOut (double\* softOut);

##### **Parameters**

Name	Direction	Description
softOut	out	The border outer softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The softOut parameter is invalid.

#### 5.2.14.16 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessOut** method

The **SetBorderSoftnessOut** method sets the border outer softness value.

##### **Syntax**

**HRESULT**            SetBorderSoftnessOut (double softOut);

##### **Parameters**

Name	Direction	Description
softOut	in	The desired border outer softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.17 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessIn** method

The **GetBorderSoftnessIn** method gets the border inner softness value.

##### **Syntax**

```
HRESULT GetBorderSoftnessIn (double* softIn);
```

##### **Parameters**

Name	Direction	Description
softIn	out	The border inner softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The softIn parameter is invalid.

#### 5.2.14.18 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessIn** method

The **SetBorderSoftnessIn** method sets the border inner softness value.

##### **Syntax**

```
HRESULT SetBorderSoftnessIn (double softIn);
```

##### **Parameters**

Name	Direction	Description
softIn	in	The desired border inner softness value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.14.19 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelSoftness** method

The **GetBorderBevelSoftness** method gets the border bevel softness value.

**Syntax**

```
HRESULT      GetBorderBevelSoftness (double* bevelSoft);
```

**Parameters**

Name	Direction	Description
bevelSoft	out	The border bevel softness value.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The bevelSoft parameter is invalid.

5.2.14.20 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelSoftness** method

The **SetBorderBevelSoftness** method sets the border bevel softness value.

**Syntax**

```
HRESULT      SetBorderBevelSoftness (double bevelSoft);
```

**Parameters**

Name	Direction	Description
bevelSoft	in	The desired border bevel softness value.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.14.21 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelPosition** method

The **GetBorderBevelPosition** method gets the border bevel position value.

##### **Syntax**

```
HRESULT GetBorderBevelPosition (double* bevelPosition);
```

##### **Parameters**

Name	Direction	Description
bevelPosition	out	The border bevel position value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The bevelPosition parameter is invalid.

#### 5.2.14.22 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelPosition** method

The **SetBorderBevelPosition** method sets the border bevel position value.

##### **Syntax**

```
HRESULT SetBorderBevelPosition (double bevelPosition);
```

##### **Parameters**

Name	Direction	Description
bevelPosition	in	The desired border bevel position value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.23 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderOpacity** method

The **GetBorderOpacity** method gets the border opacity value.

##### **Syntax**

**HRESULT**            **GetBorderOpacity** (double\* opacity);

##### **Parameters**

Name	Direction	Description
opacity	out	The border opacity value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The opacity parameter is invalid.

#### 5.2.14.24 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderOpacity** method

The **SetBorderOpacity** method sets the border opacity value.

##### **Syntax**

**HRESULT**            **SetBorderOpacity** (double opacity);

##### **Parameters**

Name	Direction	Description
opacity	in	The desired border opacity value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.25 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderHue** method

The **GetBorderHue** method gets the border hue value.

##### **Syntax**

```
HRESULT GetBorderHue (double* hue);
```

##### **Parameters**

Name	Direction	Description
hue	out	The border hue value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The hue parameter is invalid.

#### 5.2.14.26 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderHue** method

The **SetBorderHue** method sets the border hue value.

##### **Syntax**

```
HRESULT SetBorderHue (double hue);
```

##### **Parameters**

Name	Direction	Description
hue	in	The desired border hue value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.27 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSaturation** method

The **GetBorderSaturation** method gets the border saturation value.

##### **Syntax**

```
HRESULT GetBorderSaturation (double* sat);
```

##### **Parameters**

Name	Direction	Description
sat	out	The border saturation value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The sat parameter is invalid.

#### 5.2.14.28 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSaturation** method

The **SetBorderSaturation** method sets the border saturation value.

##### **Syntax**

```
HRESULT SetBorderSaturation (double sat);
```

##### **Parameters**

Name	Direction	Description
sat	in	The desired border saturation value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.29 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLuma** method

The **GetBorderLuma** method gets the border luminance value.

##### **Syntax**

```
HRESULT GetBorderLuma (double* luma);
```

##### **Parameters**

Name	Direction	Description
luma	out	The border luminance value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The luma parameter is invalid.

#### 5.2.14.30 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLuma** method

The **SetBorderLuma** method sets the border luminance value.

##### **Syntax**

```
HRESULT SetBorderLuma (double luma);
```

##### **Parameters**

Name	Direction	Description
luma	in	The desired border luminance value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.14.31 IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceDirection method

The **GetBorderLightSourceDirection** method gets the border light source direction value.

#### Syntax

```
HRESULT GetBorderLightSourceDirection (double* degrees);
```

#### Parameters

Name	Direction	Description
degrees	out	The border light source direction in degrees.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The degrees parameter is invalid.

#### 5.2.14.32 IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLightSourceDirection method

The **SetBorderLightSourceDirection** method sets the border light source direction value.

##### Syntax

**HRESULT**           SetBorderLightSourceDirection (double degrees);

##### Parameters

Name	Direction	Description
degrees	in	The desired border light source direction value in degrees.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.33 **IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceAltitude** method

The **GetBorderLightSourceAltitude** method gets the border light source altitude value.

##### **Syntax**

```
HRESULT GetBorderLightSourceAltitude (double* altitude);
```

##### **Parameters**

Name	Direction	Description
altitude	out	The border light source altitude value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The altitude parameter is invalid.

#### 5.2.14.34 **IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLightSourceAltitude** method

The **SetBorderLightSourceAltitude** method sets the border light source altitude value.

##### **Syntax**

```
HRESULT SetBorderLightSourceAltitude (double altitude);
```

##### **Parameters**

Name	Direction	Description
altitude	in	The desired border light source altitude value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 5.2.14.35 **IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskTop** method

The **GetMaskTop** method returns the mask top value.

##### **Syntax**

```
HRESULT GetMaskTop (double* maskTop);
```

##### **Parameters**

Name	Direction	Description
maskTop	out	The mask top value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskTop parameter is invalid.

#### 5.2.14.36 **IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskTop** method

The **SetMaskTop** method sets the mask top value.

##### **Syntax**

```
HRESULT SetMaskTop (double maskTop);
```

##### **Parameters**

Name	Direction	Description
maskTop	in	The desired mask top value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.37 **IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskBottom** method

The **GetMaskBottom** method returns the mask bottom value.

##### **Syntax**

**HRESULT**            **GetMaskBottom** (double\* maskBottom);

##### **Parameters**

Name	Direction	Description
maskBottom	out	The mask bottom value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskBottom parameter is invalid.

#### 5.2.14.38 **IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskBottom** method

The **SetMaskBottom** method sets the mask bottom value.

##### **Syntax**

**HRESULT**            **SetMaskBottom** (double maskBottom);

##### **Parameters**

Name	Direction	Description
maskBottom	in	The desired mask bottom value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.39 **IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskLeft** method

The **GetMaskLeft** method returns the mask left value.

##### **Syntax**

**HRESULT** GetMaskLeft (double\* maskLeft);

##### **Parameters**

Name	Direction	Description
maskLeft	out	The mask left value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskLeft parameter is invalid.

#### 5.2.14.40 **IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskLeft** method

The **SetMaskLeft** method sets the mask left value.

##### **Syntax**

**HRESULT** SetMaskLeft (double maskLeft);

##### **Parameters**

Name	Direction	Description
maskLeft	in	The desired mask left value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.41 **IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskRight** method

The **GetMaskRight** method returns the mask right value.

##### **Syntax**

**HRESULT** GetMaskRight (double\* maskRight);

##### **Parameters**

Name	Direction	Description
maskRight	out	The mask right value.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The maskRight parameter is invalid.

#### 5.2.14.42 **IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskRight** method

The **SetMaskRight** method sets the mask right value.

##### **Syntax**

**HRESULT** SetMaskRight (double maskRight);

##### **Parameters**

Name	Direction	Description
maskRight	in	The desired mask right value.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.14.43 IBMDSwitcherKeyFlyKeyFrameParameters::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyFlyKeyFrameParameters** object. Pass an object implementing the **IBMDSwitcherKeyFlyKeyFrameParametersCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

```
HRESULT AddCallback (IBMDSwitcherKeyFlyKeyFrameParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyFlyKeyFrameParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 5.2.14.44 IBMDSwitcherKeyFlyKeyFrameParameters::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherKeyFlyKeyFrameParametersCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherKeyFlyKeyFrameParametersCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.15 **IBMDSwitcherKeyFlyKeyFrameParametersCallback Interface**

The **IBMDSwitcherKeyFlyKeyFrameParametersCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherKeyFlyKeyFrameParameters** object. Like all callback methods, these callback methods may be called from another thread.

**Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherKeyFlyKeyFrameParameters	IID_ IBMDSwitcherKeyFlyKeyFrame Parameters	An <b>IBMDSwitcherKeyFlyKeyFrameParametersCallback</b> object interface is installed with <b>IBMDSwitcherKeyFlyKeyFrameParameters::AddCallback</b> and removed with <b>IBMDSwitcherKeyFlyKeyFrameParameters::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 5.2.15.1 IBMDSwitcherKeyFlyKeyFrameParametersCallback::Notify method

The Notify method is called when **IBMDSwitcherKeyFlyKeyFrameParameters** events occur, events such as a property change.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT** Notify (BMDSwitcherKeyFlyKeyFrameParametersEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherKeyFlyKeyFrameParametersEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



5.2.16 IBMDSwitcherDownstreamKeylterator Interface

The **IBMDSwitcherDownstreamKeylterator** is used to enumerate the available downstream keys.

A reference to an **IBMDSwitcherDownstreamKeylterator** object interface may be obtained from an **IBMDSwitcher** object interface using the **Createrlterator** method. Pass **IID\_IBMDSwitcherDownstreamKeylterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::Createrlterator</b> can return an <b>IBMDSwitcherDownstreamKeylterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherDownstreamKey</b> object interface.

### 5.2.16.1 IBMDSwitcherDownstreamKeyIterator::Next method

The **Next** method returns the next available **IBMDSwitcherDownstreamKey** object interface.

#### Syntax

```
HRESULT      Next (IBMDSwitcherDownstreamKey** downstreamKey);
```

#### Parameters

Name	Direction	Description
downstreamKey	out	<b>IBMDSwitcherDownstreamKey</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherDownstreamKey</b> objects available.
E_POINTER	The downstreamKey parameter is invalid.

### 5.2.17 IBMDSwitcherDownstreamKey Interface

The **IBMDSwitcherDownstreamKey** object interface is used for managing the settings of a downstream key.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherDownstreamKeyIterator	IID_IBMDSwitcherDownstreamKeyIterator	An <b>IBMDSwitcherDownstreamKey</b> object will be returned after a successful call to <b>IBMDSwitcherDownstreamKeylterator::Next</b> method.

Public Member Functions	
Method	Description
GetInputCut	Get the current cut input source.
SetInputCut	Set the cut input source.
GetInputFill	Get the current fill input source.
SetInputFill	Set the fill input source.
GetFillInputAvailabilityMask	Get the availability mask for the fill of this input.
GetCutInputAvailabilityMask	Get the availability mask for the cut of this input.
GetTie	Get the current tie flag.
SetTie	Set the tie flag.
GetRate	Get the current rate value.
SetRate	Set the rate value.
GetOnAir	Get the current on-air flag.
SetOnAir	Set the on-air flag.
PerformAutoTransition	Perform an auto-transition.
IsTransitioning	Determines if this downstream key is transitioning.
IsAutoTransitioning	Determines if this downstream key is auto-transitioning.

Public Member Functions	
Method	Description
GetFramesRemaining	Get the number of frames remaining in the transition.
GetPreMultiplied	Get the current pre-multiplied flag.
SetPreMultiplied	Set the pre-multiplied flag.
GetClip	Get the current clip value.
SetClip	Set the clip value.
GetGain	Get the current gain value.
SetGain	Set the gain value.
GetInverse	Get the current inverse flag.
SetInverse	Set the inverse flag.
GetMasked	Get the current masked flag.
SetMasked	Set the masked flag.
GetMaskTop	Get the current mask top value.
SetMaskTop	Set the mask top value.
GetMaskBottom	Get the current mask bottom value.
SetMaskBottom	Set the mask bottom value.
GetMaskLeft	Get the current mask left value.
SetMaskLeft	Set the mask left value.
GetMaskRight	Get the current mask right value.
SetMaskRight	Set the mask right value.
ResetMask	Reset mask properties to default.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

### 5.2.17.1 IBMDSwitcherDownstreamKey::GetInputCut method

The **GetInputCut** method returns the selected cut input source.

#### Syntax

```
HRESULT      GetInputCut (BMDSwitcherInputId* inputId);
```

#### Parameters

Name	Direction	Description
inputId	out	<b>BMDSwitcherInputId</b> of the selected cut input source.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inputId parameter is invalid.

### 5.2.17.2 IBMDSwitcherDownstreamKey::SetInputCut method

The **SetInputCut** method sets the cut input source.

#### Syntax

```
HRESULT      SetInputCut (BMDSwitcherInputId inputId);
```

#### Parameters

Name	Direction	Description
inputId	in	The desired cut input source's <b>BMDSwitcherInputId</b> .

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The inputId parameter is invalid.

### 5.2.17.3 **IBMDSwitcherDownstreamKey::GetInputFill** method

The **GetInputFill** method returns the selected fill input source.

#### **Syntax**

**HRESULT**            **GetInputFill** (BMDSwitcherInputId\* inputId);

#### **Parameters**

Name	Direction	Description
inputId	out	<b>BMDSwitcherInputId</b> of the selected fill input source.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The inputId parameter is invalid.

#### 5.2.17.4 IBMDSwitcherDownstreamKey::SetInputFill method

The **SetInputFill** method sets the fill input source.

##### Syntax

```
HRESULT      SetInputFill (BMDSwitcherInputId inputId);
```

##### Parameters

Name	Direction	Description
inputId	in	The desired fill input source's <b>BMDSwitcherInputId</b> .

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The inputId parameter is invalid.



### 5.2.17.5 IBMDSwitcherDownstreamKey::GetFillInputAvailabilityMask method

The GetFillInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for fill inputs available to this downstream key. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a fill input for this downstream key.

#### Syntax

**HRESULT**            GetFillInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

### 5.2.17.6 IBMDSwitcherDownstreamKey::GetCutInputAvailabilityMask method

The GetCutInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for cut inputs available to this downstream key. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a cut input for this downstream key.

#### Syntax

**HRESULT**            GetCutInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

#### 5.2.17.7 **IBMDSwitcherDownstreamKey::GetTie** method

The **GetTie** method gets the current tie flag.

##### **Syntax**

```
HRESULT GetTie (boolean* tie);
```

##### **Parameters**

Name	Direction	Description
tie	out	Boolean tie flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The tie parameter is invalid.

#### 5.2.17.8 **IBMDSwitcherDownstreamKey::SetTie** method

The **SetTie** method sets the tie flag.

##### **Syntax**

```
HRESULT SetTie (boolean tie);
```

##### **Parameters**

Name	Direction	Description
tie	in	The desired tie flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.9 IBMDSwitcherDownstreamKey::GetRate method

The **GetRate** method gets the current rate value.

#### Syntax

```
HRESULT          GetRate (uint32_t* frames);
```

#### Parameters

Name	Direction	Description
frames	out	The current rate value in frames.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The frames parameter is invalid.

### 5.2.17.10 IBMDSwitcherDownstreamKey::SetRate method

The **SetRate** method sets the rate value.

#### Syntax

```
HRESULT          SetRate (uint32_t frames);
```

#### Parameters

Name	Direction	Description
frames	in	The desired rate value in frames.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The frames parameter is invalid.
E_FAIL	Failure.

#### 5.2.17.11 **IBMDSwitcherDownstreamKey::GetOnAir** method

The **GetOnAir** method returns the on-air flag.

##### **Syntax**

```
HRESULT GetOnAir (boolean* onAir);
```

##### **Parameters**

Name	Direction	Description
onAir	out	Boolean on-air flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The onAir parameter is invalid.

#### 5.2.17.12 **IBMDSwitcherDownstreamKey::SetOnAir** method

The **SetOnAir** method sets the on-air flag.

##### **Syntax**

```
HRESULT SetOnAir (boolean onAir);
```

##### **Parameters**

Name	Direction	Description
onAir	in	The desired on-air flag.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.17.13 **IBMDSwitcherDownstreamKey::PerformAutoTransition** method

Use the **PerformAutoTransition** method to start an auto-transition.

##### **Syntax**

```
HRESULT PerformAutoTransition (void);
```

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 5.2.17.14 **IBMDSwitcherDownstreamKey::IsTransitioning** method

The **IsTransitioning** method returns whether this downstream key is transitioning or not.

##### **Syntax**

```
HRESULT IsTransitioning (boolean* isTransitioning);
```

##### **Parameters**

Name	Direction	Description
isTransitioning	out	Boolean status of whether it is transitioning.

##### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The isTransitioning parameter is invalid.

### 5.2.17.15 **IBMDSwitcherDownstreamKey::IsAutoTransitioning** method

The **IsAutoTransitioning** method returns whether this downstream key is auto-transitioning or not.

#### Syntax

```
HRESULT IsAutoTransitioning (boolean* isAutoTransitioning);
```

#### Parameters

Name	Direction	Description
isAutoTransitioning	out	Boolean status of whether it is auto-transitioning.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The isAutoTransitioning parameter is invalid.

### 5.2.17.16 **IBMDSwitcherDownstreamKey::GetFramesRemaining** method

The **GetFramesRemaining** method gets the number of frames remaining in the transition.

#### Syntax

```
HRESULT GetFramesRemaining (uint32_t* framesRemaining);
```

#### Parameters

Name	Direction	Description
framesRemaining	out	Number of frames remaining in the transition.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The framesRemaining parameter is invalid.



### 5.2.17.17 IBMDSwitcherDownstreamKey::GetPreMultiplied method

The **GetPreMultiplied** method returns the current pre-multiplied flag.

#### Syntax

```
HRESULT      GetPreMultiplied (boolean* preMultiplied);
```

#### Parameters

Name	Direction	Description
preMultiplied	out	The current pre-multiplied flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The preMultiplied parameter is invalid.

### 5.2.17.18 IBMDSwitcherDownstreamKey::SetPreMultiplied method

The **SetPreMultiplied** method sets the pre-multiplied flag. Note that clip, gain and inverse controls are not used when pre-multiplied flag is set to true.

#### Syntax

```
HRESULT SetPreMultiplied (boolean preMultiplied);
```

#### Parameters

Name	Direction	Description
preMultiplied	in	The desired pre-multiplied flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.19 **IBMDSwitcherDownstreamKey::GetClip method**

The **GetClip** method returns the current clip value.

#### **Syntax**

```
HRESULT GetClip (double* clip);
```

#### **Parameters**

Name	Direction	Description
clip	out	The current clip value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The clip parameter is invalid.

### 5.2.17.20 **IBMDSwitcherDownstreamKey::SetClip method**

The **SetClip** method sets the clip value.

#### **Syntax**

```
HRESULT SetClip (double clip);
```

#### **Parameters**

Name	Direction	Description
clip	in	The desired clip value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.21 **IBMDSwitcherDownstreamKey::GetGain** method

The **GetGain** method returns the current gain value.

#### **Syntax**

```
HRESULT GetGain (double* gain);
```

#### **Parameters**

Name	Direction	Description
gain	out	The current gain value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

### 5.2.17.22 **IBMDSwitcherDownstreamKey::SetGain** method

The **SetGain** method sets the gain value.

#### **Syntax**

```
HRESULT SetGain (double gain);
```

#### **Parameters**

Name	Direction	Description
gain	in	The desired gain value.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.23 **IBMDSwitcherDownstreamKey::GetInverse** method

The **GetInverse** method returns the current inverse flag.

#### **Syntax**

```
HRESULT GetInverse (boolean* inverse);
```

#### **Parameters**

Name	Direction	Description
inverse	out	The current inverse flag.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

### 5.2.17.24 **IBMDSwitcherDownstreamKey::SetInverse** method

The **SetInverse** method sets the inverse flag.

#### **Syntax**

```
HRESULT SetInverse (boolean inverse);
```

#### **Parameters**

Name	Direction	Description
inverse	in	The desired inverse flag.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.25 **IBMDSwitcherDownstreamKey::GetMasked** method

The **GetMasked** method returns whether masking is enabled or not.

#### Syntax

```
HRESULT GetMasked (boolean* masked);
```

#### Parameters

Name	Direction	Description
masked	out	Boolean flag of whether masking is enabled.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The masked parameter is invalid.

### 5.2.17.26 **IBMDSwitcherDownstreamKey::SetMasked** method

The **SetMasked** method enables or disables masking.

#### Syntax

```
HRESULT SetMasked (boolean masked);
```

#### Parameters

Name	Direction	Description
masked	in	The desired masked value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.27 **IBMDSwitcherDownstreamKey::GetMaskTop** method

The **GetMaskTop** method returns the current mask top value.

#### Syntax

```
HRESULT GetMaskTop (double* maskTop);
```

#### Parameters

Name	Direction	Description
maskTop	out	The current mask top value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskTop parameter is invalid.

### 5.2.17.28 **IBMDSwitcherDownstreamKey::SetMaskTop** method

The **SetMaskTop** method sets the mask top value.

#### Syntax

```
HRESULT SetMaskTop (double maskTop);
```

#### Parameters

Name	Direction	Description
maskTop	in	The desired mask top value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.29 **IBMDSwitcherDownstreamKey::GetMaskBottom** method

The **GetMaskBottom** method returns the current mask bottom value.

#### Syntax

```
HRESULT GetMaskBottom (double* maskBottom);
```

#### Parameters

Name	Direction	Description
maskBottom	out	The current mask bottom value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskBottom parameter is invalid.

### 5.2.17.30 **IBMDSwitcherDownstreamKey::SetMaskBottom** method

The **SetMaskBottom** method sets the mask bottom value.

#### Syntax

```
HRESULT SetMaskBottom (double maskBottom);
```

#### Parameters

Name	Direction	Description
maskBottom	in	The desired mask bottom value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 5.2.17.31 **IBMDSwitcherDownstreamKey::GetMaskLeft** method

The **GetMaskLeft** method returns the current mask left value.

#### Syntax

```
HRESULT GetMaskLeft (double* maskLeft);
```

#### Parameters

Name	Direction	Description
maskLeft	out	The current mask left value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskLeft parameter is invalid.

### 5.2.17.32 **IBMDSwitcherDownstreamKey::SetMaskLeft** method

The **SetMaskLeft** method sets the mask left value.

#### Syntax

```
HRESULT SetMaskLeft (double maskLeft);
```

#### Parameters

Name	Direction	Description
maskLeft	in	The desired mask left value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.33 **IBMDSwitcherDownstreamKey::GetMaskRight** method

The **GetMaskRight** method returns the current mask right value.

#### Syntax

```
HRESULT GetMaskRight (double* maskRight);
```

#### Parameters

Name	Direction	Description
maskRight	out	The current mask right value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The maskRight parameter is invalid.

### 5.2.17.34 **IBMDSwitcherDownstreamKey::SetMaskRight** method

The **SetMaskRight** method sets the mask right value.

#### Syntax

```
HRESULT SetMaskRight (double maskRight);
```

#### Parameters

Name	Direction	Description
maskRight	in	The desired mask right value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.35 IBMDSwitcherDownstreamKey::ResetMask method

The **ResetMask** method resets mask settings to the default values.

#### Syntax

```
HRESULT      ResetMask (void);
```

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 5.2.17.36 IBMDSwitcherDownstreamKey::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherDownstreamKey** object. Pass an object implementing the **IBMDSwitcherDownstreamKeyCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

**HRESULT** AddCallback (IBMDSwitcherDownstreamKeyCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherDownstreamKeyCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 5.2.17.37 IBMDSwitcherDownstreamKey::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherDownstreamKeyCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherDownstreamKeyCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

5.2.18

IBMDSwitcherDownstreamKeyCallback Interface

The **IBMDSwitcherDownstreamKeyCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherDownstreamKey** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherDownstreamKey	IID_IBMDSwitcherDownstreamKey	An <b>IBMDSwitcherDownstreamKeyCallback</b> object interface is installed with <b>IBMDSwitcherDownstreamKey::AddCallback</b> and removed with <b>IBMDSwitcherDownstreamKey::RemoveCallback</b>

Public Member Functions	
Method	Description
Next	Called when an event occurs.

### 5.2.18.1 **IBMDSwitcherDownstreamKeyCallback::Notify**

The **Notify** method is called when **IBMDSwitcherDownstreamKey** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **Notify** (BMDSwitcherDownstreamKeyEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherDownstreamKeyEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 6 SuperSource

When available in the target switcher, the SuperSource allows multiple video sources to be displayed within boxes as part of a single video output.

### 6.1 SuperSource Data Types

#### 6.1.1 SuperSource Box Event Type

**BMDSwitcherSuperSourceBoxEventType** enumerates the possible event types for **BMDSwitcherSuperSourceBox**.

<b>bmdSwitcherSuperSourceBoxEventTypeInputSourceChanged</b>	The source input changed.
<b>bmdSwitcherSuperSourceBoxEventTypePositionXChanged</b>	The x position changed.
<b>bmdSwitcherSuperSourceBoxEventTypePositionYChanged</b>	The y position changed.
<b>bmdSwitcherSuperSourceBoxEventTypeSizeChanged</b>	The size changed.
<b>bmdSwitcherSuperSourceBoxEventTypeCroppedChanged</b>	The cropped flag changed.
<b>bmdSwitcherSuperSourceBoxEventTypeCropTopChanged</b>	The top crop value changed.
<b>bmdSwitcherSuperSourceBoxEventTypeCropBottomChanged</b>	The bottom crop value changed.
<b>bmdSwitcherSuperSourceBoxEventTypeCropLeftChanged</b>	The left crop value changed.
<b>bmdSwitcherSuperSourceBoxEventTypeCropRightChanged</b>	The right crop value changed.



## 6.1.2 SuperSource Input Event Type

**BMDSwitcherInputSuperSourceEventType** enumerates the possible event types for **BMDSwitcherInputSuperSource**.

<b>bmdSwitcherInputSuperSourceEventTypeInputFillChanged</b>	The fill input changed.
<b>bmdSwitcherInputSuperSourceEventTypeInputCutChanged</b>	The cut input changed.
<b>bmdSwitcherInputSuperSourceEventTypeArtOptionChanged</b>	The art option changed.
<b>bmdSwitcherInputSuperSourceEventTypePreMultipliedChanged</b>	The pre-multiplied flag changed.
<b>bmdSwitcherInputSuperSourceEventTypeClipChanged</b>	The clip value changed.
<b>bmdSwitcherInputSuperSourceEventTypeGainChanged</b>	The gain changed.
<b>bmdSwitcherInputSuperSourceEventTypeInverseChanged</b>	The inverse flag changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderEnabledChanged</b>	The border enabled flag changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderBevelChanged</b>	The border bevel changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderWidthOutChanged</b>	The border outer width changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderWidthInChanged</b>	The border inner width changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderSoftnessOutChanged</b>	The border outer softness changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderSoftnessInChanged</b>	The border inner softness changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderBevelSoftnessChanged</b>	The border bevel softness changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderBevelPositionChanged</b>	The border bevel position changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderHueChanged</b>	The border hue changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderSaturationChanged</b>	The border saturation changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderLumaChanged</b>	The border luminescence changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderLightSourceDirectionChanged</b>	The border light source direction changed.
<b>bmdSwitcherInputSuperSourceEventTypeBorderLightSourceAltitudeChanged</b>	The border light source altitude changed.

### 6.1.3 SuperSource Art Option

**BMDSwitcherSuperSourceArtOption** enumerates the possible supersource art options, used by the **IBMDSwitcherInputSuperSource** object interface.

**bmdSwitcherSuperSourceArtOptionBackground**  
**bmdSwitcherSuperSourceArtOptionForeground**

Places art in the background.  
Places art in the foreground.

6.2

Interface Reference

6.2.1

IBMDSwitcherInputSuperSource Interface

The **IBMDSwitcherInputSuperSource** object interface is used for manipulating settings specific to the SuperSource input.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInput	IID_IBMDSwitcherInput	An <b>IBMDSwitcherInputSuperSource</b> object interface can be obtained with <b>IBMDSwitcherInput::QueryInterface</b> .

Public Member Functions	
Method	Description
GetInputCut	Get the current art cut input.
SetInputCut	Set the art cut input.
GetInputFill	Get the current art fill input.
SetInputFill	Set the art fill input.
GetFillInputAvailabilityMask	Get the availability mask for the fill of this input.
GetCutInputAvailabilityMask	Get the availability mask for the cut of this input.
GetArtOption	Get the current art option.
SetArtOption	Set the art option.
GetPreMultiplied	Get the current art pre-multiplied flag.
SetPreMultiplied	Set the art pre-multiplied flag.
GetClip	Get the current art clip value.
SetClip	Set the art clip value.
GetGain	Get the current art gain.

Public Member Functions	
Method	Description
SetGain	Set the art gain.
GetInverse	Get the current art inverse flag.
SetInverse	Set the art inverse flag.
GetBorderEnabled	Get the current border enabled flag.
SetBorderEnabled	Set the border enabled flag.
GetBorderBevel	Get the current border bevel.
SetBorderBevel	Set the border bevel.
GetBorderWidthOut	Get the current border outer width.
SetBorderWidthOut	Set the border outer width.
GetBorderWidthIn	Get the current border inner width.
SetBorderWidthIn	Set the border inner width.
GetBorderSoftnessOut	Get the current border outer softness.
SetBorderSoftnessOut	Set the border outer softness.
GetBorderSoftnessIn	Get the current border inner softness.
SetBorderSoftnessIn	Set the border inner softness.
GetBorderBevelSoftness	Get the current border bevel softness.
SetBorderBevelSoftness	Set the border bevel softness.
GetBorderBevelPosition	Get the current border bevel position.
SetBorderBevelPosition	Set the border bevel position.
GetBorderHue	Get the current border hue.
SetBorderHue	Set the border hue.
GetBorderSaturation	Get the current border saturation.
SetBorderSaturation	Set the border saturation.

Public Member Functions	
Method	Description
GetBorderLuma	Get the current border luminescence.
SetBorderLuma	Set the border luminescence.
GetBorderLightSourceDirection	Get the current border light source direction.
SetBorderLightSourceDirection	Set the border light source direction.
GetBorderLightSourceAltitude	Get the current border light source altitude.
SetBorderLightSourceAltitude	Set the border light source altitude.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.
CreateIterator	Creates an iterator.

### 6.2.1.1 IBMDSwitcherInputSuperSource::GetInputCut method

The **GetInputCut** method returns the current art cut input.

#### Syntax

```
HRESULT      GetInputCut (BMDSwitcherInputId* input);
```

#### Parameters

Name	Direction	Description
input	out	The current cut input.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

### 6.2.1.2 IBMDSwitcherInputSuperSource::SetInputCut method

The **SetInputCut** method sets the art cut input.

#### Syntax

```
HRESULT          SetInputCut (BMDSwitcherInputId input);
```

#### Parameters

Name	Direction	Description
input	in	The desired cut input.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

### 6.2.1.3 IBMDSwitcherInputSuperSource::GetInputFill method

The **GetInputFill** method returns the current art fill input.

#### Syntax

```
HRESULT      GetInputFill (BMDSwitcherInputId* input);
```

#### Parameters

Name	Direction	Description
input	out	The current fill input.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.



#### 6.2.1.4 IBMDSwitcherInputSuperSource::SetInputFill method

The **SetInputFill** method sets the art fill input.

##### Syntax

```
HRESULT          SetInputFill (BMDSwitcherInputId input);
```

##### Parameters

Name	Direction	Description
input	in	The desired fill input.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

### 6.2.1.5 IBMDSwitcherInputSuperSource::GetFillInputAvailabilityMask method

The GetFillInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for fill inputs available to this supersource input. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a fill input for this supersource.

#### Syntax

**HRESULT** GetFillInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

### 6.2.1.6 IBMDSwitcherInputSuperSource::GetCutInputAvailabilityMask method

The GetCutInputAvailabilityMask method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for cut inputs available to this supersource input. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value. If the result of the bitwise-AND is equal to the mask value then this input is available for use as a cut input for this supersource.

#### Syntax

**HRESULT** GetCutInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

#### Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

### 6.2.1.7 IBMDSwitcherInputSuperSource::GetArtOption method

The **GetArtOption** method returns the current art option.

#### Syntax

```
HRESULT      GetArtOption (BMDSwitcherSuperSourceArtOption* artOption);
```

#### Parameters

Name	Direction	Description
artOption	out	The current art option.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The artOption parameter is invalid.

### 6.2.1.8 IBMDSwitcherInputSuperSource::SetArtOption method

The **SetArtOption** method sets the art option.

#### Syntax

```
HRESULT      SetArtOption (BMDSwitcherSuperSourceArtOption artOption);
```

#### Parameters

Name	Direction	Description
artOption	in	The desired art option.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The artOption parameter is invalid.

### 6.2.1.9 IBMDSwitcherInputSuperSource::GetPreMultiplied method

The **GetPreMultiplied** method returns the current art pre-multiplied flag.

#### Syntax

```
HRESULT      GetPreMultiplied (boolean* preMultiplied);
```

#### Parameters

Name	Direction	Description
preMultiplied	out	The current pre-multiplied flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The preMultiplied parameter is invalid.

#### 6.2.1.10 IBMDSwitcherInputSuperSource::SetPreMultiplied method

The **SetPreMultiplied** method sets the art pre-multiplied flag.

##### Syntax

```
HRESULT      SetPreMultiplied (boolean preMultiplied);
```

##### Parameters

Name	Direction	Description
preMultiplied	in	The desired pre-multiplied flag.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The preMultiplied parameter is invalid.

6.2.1.11 **IBMDSwitcherInputSuperSource::GetClip** method

The **GetClip** method returns the current art clip value.

**Syntax**

```
HRESULT          GetClip (double* clip);
```

**Parameters**

Name	Direction	Description
clip	out	The current clip value.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The clip parameter is invalid.

6.2.1.12 **IBMDSwitcherInputSuperSource::SetClip** method

The **SetClip** method sets the art clip value.

**Syntax**

```
HRESULT          SetClip (double clip);
```

**Parameters**

Name	Direction	Description
clip	in	The desired clip value.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.



### 6.2.1.13 IBMDSwitcherInputSuperSource::GetGain method

The **GetGain** method returns the current art gain.

#### Syntax

```
HRESULT      GetGain (double* gain);
```

#### Parameters

Name	Direction	Description
gain	out	The current gain.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

### 6.2.1.14 IBMDSwitcherInputSuperSource::SetGain method

The **SetGain** method sets the art gain.

#### Syntax

```
HRESULT      SetGain (double gain);
```

#### Parameters

Name	Direction	Description
gain	in	The desired gain.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.1.15 IBMDSwitcherInputSuperSource::GetInverse method

The **GetInverse** method returns the current art inverse flag.

##### Syntax

```
HRESULT      GetInverse (boolean* inverse);
```

##### Parameters

Name	Direction	Description
inverse	out	The current inverse flag.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The inverse parameter is invalid.

#### 6.2.1.16 IBMDSwitcherInputSuperSource::SetInverse method

The **SetInverse** method sets the art inverse flag.

##### Syntax

```
HRESULT      SetInverse (boolean inverse);
```

##### Parameters

Name	Direction	Description
inverse	in	The desired inverse flag.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.1.17 IBMDSwitcherInputSuperSource::GetBorderEnabled method

The **GetBorderEnabled** method returns the current border enabled flag.

##### Syntax

**HRESULT** GetBorderEnabled (boolean\* enabled);

##### Parameters

Name	Direction	Description
enabled	out	The current border enabled flag.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The enabled parameter is invalid.

#### 6.2.1.18 IBMDSwitcherInputSuperSource::SetBorderEnabled method

The **SetBorderEnabled** method sets the border enabled flag.

##### Syntax

**HRESULT** SetBorderEnabled (boolean enabled);

##### Parameters

Name	Direction	Description
enabled	in	The desired border enabled flag

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.19 IBMDSwitcherInputSuperSource::GetBorderBevel method

The **GetBorderBevel** method returns the current border bevel option.

#### Syntax

```
HRESULT      GetBorderBevel (BMDSwitcherBorderBevelOption* bevelOption);
```

#### Parameters

Name	Direction	Description
bevelOption	out	The current border bevel option.

#### Return Values

Value	Description
S_OK	Success.
E_UNEXPECTED	Unexpected error occurred.
E_POINTER	The bevelOption parameter is invalid.

### 6.2.1.20 IBMDSwitcherInputSuperSource::SetBorderBevel method

The **SetBorderBevel** method sets the border bevel option.

#### Syntax

```
HRESULT          SetBorderBevel (BMDSwitcherBorderBevelOption bevelOption);
```

#### Parameters

Name	Direction	Description
bevelOption	in	The desired border bevel option.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The bevelOption parameter is invalid.
E_FAIL	Failure.

#### 6.2.1.21 IBMDSwitcherInputSuperSource::GetBorderWidthOut method

The **GetBorderWidthOut** method returns the current border outer width.

##### Syntax

```
HRESULT GetBorderWidthOut (double* widthOut);
```

##### Parameters

Name	Direction	Description
widthOut	out	The current border outer width.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The widthOut parameter is invalid.

#### 6.2.1.22 IBMDSwitcherInputSuperSource::SetBorderWidthOut method

The **SetBorderWidthOut** method sets the border outer width.

##### Syntax

```
HRESULT SetBorderWidthOut (double widthOut);
```

##### Parameters

Name	Direction	Description
widthOut	in	The desired border outer width.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.1.23 IBMDSwitcherInputSuperSource::GetBorderWidthIn method

The **GetBorderWidthIn** method returns the current border inner width.

##### Syntax

```
HRESULT GetBorderWidthIn (double* widthIn);
```

##### Parameters

Name	Direction	Description
widthIn	out	The current border inner width.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The widthIn parameter is invalid.

#### 6.2.1.24 IBMDSwitcherInputSuperSource::SetBorderWidthIn method

The **SetBorderWidthIn** method sets the border inner width.

##### Syntax

```
HRESULT SetBorderWidthIn (double widthIn);
```

##### Parameters

Name	Direction	Description
widthIn	in	The desired border inner width.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.1.25 IBMDSwitcherInputSuperSource::GetBorderSoftnessOut method

The **GetBorderSoftnessOut** method returns the current border outer softness.

##### Syntax

```
HRESULT GetBorderSoftnessOut (double* softnessOut);
```

##### Parameters

Name	Direction	Description
softnessOut	out	The current border outer softness.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The softnessOut parameter is invalid.

#### 6.2.1.26 IBMDSwitcherInputSuperSource::SetBorderSoftnessOut method

The **SetBorderSoftnessOut** method sets the border outer softness.

##### Syntax

```
HRESULT SetBorderSoftnessOut (double softnessOut);
```

##### Parameters

Name	Direction	Description
softnessOut	in	The desired border outer softness.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 6.2.1.27 IBMDSwitcherInputSuperSource::GetBorderSoftnessIn method

The **GetBorderSoftnessIn** method returns the current border inner softness.

##### Syntax

```
HRESULT      GetBorderSoftnessIn (double* softnessIn);
```

##### Parameters

Name	Direction	Description
softnessIn	out	The current border inner softness.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The softnessIn parameter is invalid.

#### 6.2.1.28 IBMDSwitcherInputSuperSource::SetBorderSoftnessIn method

The **SetBorderSoftnessIn** method sets the border inner softness.

##### Syntax

```
HRESULT      SetBorderSoftnessIn (double softnessIn);
```

##### Parameters

Name	Direction	Description
softnessIn	in	The desired border inner softness.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.29 IBMDSwitcherInputSuperSource::GetBorderBevelSoftness method

The **GetBorderBevelSoftness** method returns the current border bevel softness.

#### Syntax

```
HRESULT GetBorderBevelSoftness (double* bevelSoftness);
```

#### Parameters

Name	Direction	Description
bevelSoftness	out	The current border bevel softness.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The bevelSoftness parameter is invalid.

### 6.2.1.30 IBMDSwitcherInputSuperSource::SetBorderBevelSoftness method

The **SetBorderBevelSoftness** method sets the border bevel softness.

#### Syntax

```
HRESULT SetBorderBevelSoftness (double bevelSoftness);
```

#### Parameters

Name	Direction	Description
bevelSoftness	in	The desired border bevel softness.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.31 IBMDSwitcherInputSuperSource::GetBorderBevelPosition method

The **GetBorderBevelPosition** method returns the current border bevel position.

#### Syntax

```
HRESULT GetBorderBevelPosition (double* bevelPosition);
```

#### Parameters

Name	Direction	Description
bevelPosition	out	The current border bevel position.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The bevelPosition parameter is invalid.

### 6.2.1.32 IBMDSwitcherInputSuperSource::SetBorderBevelPosition method

The **SetBorderBevelPosition** method sets the border bevel position.

#### Syntax

```
HRESULT SetBorderBevelPosition (double bevelPosition);
```

#### Parameters

Name	Direction	Description
bevelPosition	in	The desired border bevel position.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.33 IBMDSwitcherInputSuperSource::GetBorderHue method

The **GetBorderHue** method returns the current border hue.

#### Syntax

```
HRESULT GetBorderHue (double* hue);
```

#### Parameters

Name	Direction	Description
hue	out	The current border hue.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The hue parameter is invalid.

### 6.2.1.34 IBMDSwitcherInputSuperSource::SetBorderHue method

The **SetBorderHue** method sets the border hue.

#### Syntax

```
HRESULT SetBorderHue (double hue);
```

#### Parameters

Name	Direction	Description
hue	in	The desired border hue.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.35 IBMDSwitcherInputSuperSource::GetBorderSaturation method

The **GetBorderSaturation** method returns the current border saturation.

#### Syntax

```
HRESULT GetBorderSaturation (double* sat);
```

#### Parameters

Name	Direction	Description
sat	out	The current border saturation.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The sat parameter is invalid.

### 6.2.1.36 IBMDSwitcherInputSuperSource::SetBorderSaturation method

The **SetBorderSaturation** method sets the border saturation.

#### Syntax

```
HRESULT SetBorderSaturation (double sat);
```

#### Parameters

Name	Direction	Description
sat	in	The desired border saturation.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.1.37 IBMDSwitcherInputSuperSource::GetBorderLuma method

The **GetBorderLuma** method returns the current border luminescence.

#### Syntax

```
HRESULT GetBorderLuma (double* luma);
```

#### Parameters

Name	Direction	Description
luma	out	The current border luminescence.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The luma parameter is invalid.

### 6.2.1.38 IBMDSwitcherInputSuperSource::SetBorderLuma method

The **SetBorderLuma** method sets the border luminescence.

#### Syntax

```
HRESULT SetBorderLuma (double luma);
```

#### Parameters

Name	Direction	Description
luma	in	The desired border luminescence.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.1.39 **IBMDSwitcherInputSuperSource::GetBorderLightSourceDirection method**

The **GetBorderLightSourceDirection** method returns the current border light source direction.

**Syntax**

```
HRESULT          GetBorderLightSourceDirection (double* degrees);
```

**Parameters**

Name	Direction	Description
degrees	out	The current border light source direction in degrees.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The degrees parameter is invalid.

6.2.1.40 **IBMDSwitcherInputSuperSource::SetBorderLightSourceDirection method**

The **SetBorderLightSourceDirection** method sets the border light source direction.

**Syntax**

```
HRESULT          SetBorderLightSourceDirection (double degrees);
```

**Parameters**

Name	Direction	Description
degrees	in	The desired border light source direction in degrees.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.1.41 **IBMDSwitcherInputSuperSource::GetBorderLightSourceAltitude** method

The **GetBorderLightSourceAltitude** method returns the current border light source altitude.

##### Syntax

```
HRESULT GetBorderLightSourceAltitude (double* altitude);
```

##### Parameters

Name	Direction	Description
altitude	out	The current border light source altitude.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The altitude parameter is invalid.

#### 6.2.1.42 **IBMDSwitcherInputSuperSource::SetBorderLightSourceAltitude** method

The **SetBorderLightSourceAltitude** method sets the border light source altitude.

##### Syntax

```
HRESULT SetBorderLightSourceAltitude (double altitude);
```

##### Parameters

Name	Direction	Description
altitude	in	The desired border light source altitude.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



6.2.1.43 **IBMDSwitcherInputSuperSource::AddCallback method**

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherInputSuperSource** object. Pass an object implementing the **IBMDSwitcherInputSuperSourceCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

**Syntax**

```
HRESULT AddCallback (IBMDSwitcherInputSuperSourceCallback* callback);
```

**Parameters**

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputSuperSourceCallback</b> object interface.

**Return Values**

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 6.2.1.44 IBMDSwitcherInputSuperSource::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherInputSuperSourceCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherInputSuperSourceCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

6.2.2

IBMDSwitcherInputSuperSourceCallback Interface

The **IBMDSwitcherInputSuperSourceCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherInputSuperSource** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputSuperSource	IID_IBMDSwitcherInputSuperSource	An <b>IBMDSwitcherInputSuperSourceCallback</b> object interface is installed with <b>IBMDSwitcherInputSuperSource::AddCallback</b> and removed with <b>IBMDSwitcherInputSuperSource::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

6.2.2.1 IBMDSwitcherInputSuperSourceCallback::Notify method

The **Notify** method is called when **IBMDSwitcherInputSuperSource** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

Syntax

```
HRESULT      Notify (BMDSwitcherInputSuperSourceEventType eventType);
```

Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherInputSuperSourceEventType</b> that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.3

IBMDSwitcherSuperSourceBoxIterator Interface

The **IBMDSwitcherSuperSourceBoxIterator** is used to enumerate the available supersource boxes for a supersource input.

A reference to an **IBMDSwitcherSuperSourceBoxIterator** object interface may be obtained from an **IBMDSwitcherInputSuperSource** object interface using the **CreateIterator** method. Pass **IID\_IBMDSwitcherSuperSourceBoxIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputSuperSource	IID_IBMDSwitcherInputSuperSource	<b>IBMDSwitcherInputSuperSource::CreateIterator</b> can return an <b>IBMDSwitcherSuperSourceBoxIterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherSuperSourceBox</b> object interface.

### 6.2.3.1 IBMDSwitcherSuperSourceBoxIterator::Next method

The **Next** method returns the next available **IBMDSwitcherSuperSourceBox** object interface.

#### Syntax

```
HRESULT         Next (IBMDSwitcherSuperSourceBox** box);
```

#### Parameters

Name	Direction	Description
box	out	<b>IBMDSwitcherSuperSourceBox</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherSuperSourceBox</b> objects available.
E_POINTER	The box parameter is invalid.

### 6.2.4 IBMDSwitcherSuperSourceBox Interface

The **IBMDSwitcherSuperSourceBox** object interface is used for manipulating supersource box settings.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSuperSourceBoxIterator	IID_IBMDSwitcherSuperSourceBoxIterator	An <b>IBMDSwitcherSuperSourceBox</b> object will be returned after a successful call to <b>IBMDSwitcherSuperSourceBoxIterator::Next</b> method.

Public Member Functions	
Method	Description
GetEnabled	Get the current enabled flag.
SetEnabled	Set the enabled flag.
GetInputSource	Get the input source.
SetInputSource	Set the input source.
GetPositionX	Get the x position.
SetPositionX	Set the x position.
GetPositionY	Get the y position.
SetPositionY	Set the y position.
GetSize	Get the size.
SetSize	Set the size.
GetCropped	Get the cropped flag.
SetCropped	Set the cropped flag.
GetCropTop	Get the top crop value.
SetCropTop	Set the top crop value.

Public Member Functions	
Method	Description
GetCropBottom	Get the bottom crop value.
SetCropBottom	Set the bottom crop value.
GetCropLeft	Get the left crop value.
SetCropLeft	Set the left crop value.
GetCropRight	Get the right crop value.
SetCropRight	Set the right crop value.
ResetCrop	Reset to default crop values.
GetInputAvailabilityMask	Get the input availability mask.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.



### 6.2.4.1 IBMDSwitcherSuperSourceBox::GetEnabled method

The **GetEnabled** method returns the current enabled flag. Enabled supersource boxes are included in the corresponding supersource input.

#### Syntax

```
HRESULT      GetEnabled (boolean* enabled);
```

#### Parameters

Name	Direction	Description
enabled	out	The current enabled flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The enabled parameter is invalid.

#### 6.2.4.2 IBMDSwitcherSuperSourceBox::SetEnabled method

The **SetEnabled** method sets the enabled flag. Enabled supersource boxes are included in the corresponding supersource input.

##### Syntax

```
HRESULT      SetEnabled (boolean enabled);
```

##### Parameters

Name	Direction	Description
enabled	in	The desired enabled flag.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 6.2.4.3 IBMDSwitcherSuperSourceBox::GetInputSource method

The **GetInputSource** method returns the current input source.

#### Syntax

```
HRESULT      GetInputSource (BMDSwitcherInputId* input);
```

#### Parameters

Name	Direction	Description
input	out	The current input source's BMDSwitcherInputId.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The input parameter is invalid.

#### 6.2.4.4 IBMDSwitcherSuperSourceBox::SetInputSource method

The **SetInputSource** method sets the input source.

##### Syntax

```
HRESULT      SetInputSource (BMDSwitcherInputId input);
```

##### Parameters

Name	Direction	Description
input	in	The desired input source's BMDSwitcherInputId.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The input parameter is invalid.

#### 6.2.4.5 IBMDSwitcherSuperSourceBox::GetPositionX method

The **GetPositionX** method returns the current x position.

##### Syntax

```
HRESULT      GetPositionX (double* positionX);
```

##### Parameters

Name	Direction	Description
positionX	out	The current x position.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The positionX parameter is invalid.

#### 6.2.4.6 IBMDSwitcherSuperSourceBox::SetPositionX method

The **SetPositionX** method sets the x position.

##### Syntax

```
HRESULT      SetPositionX (double positionX);
```

##### Parameters

Name	Direction	Description
positionX	in	The desired x position.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.7 IBMDSwitcherSuperSourceBox::GetPositionY method

The **GetPositionY** method returns the current y position.

##### Syntax

```
HRESULT GetPositionY (double* positionY);
```

##### Parameters

Name	Direction	Description
positionY	out	The current y position.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The positionY parameter is invalid.

#### 6.2.4.8 IBMDSwitcherSuperSourceBox::SetPositionY method

The **SetPositionY** method sets the y position.

##### Syntax

```
HRESULT SetPositionY (double positionY);
```

##### Parameters

Name	Direction	Description
positionY	in	The desired y position.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.9 IBMDSwitcherSuperSourceBox::GetSize method

The **GetSize** method returns the current size.

##### Syntax

```
HRESULT      GetSize (double* size);
```

##### Parameters

Name	Direction	Description
size	out	The current size.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The size parameter is invalid.

#### 6.2.4.10 IBMDSwitcherSuperSourceBox::SetSize method

The **SetSize** method sets the size.

##### Syntax

```
HRESULT      SetSize (double size);
```

##### Parameters

Name	Direction	Description
size	in	The desired size.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.11 IBMDSwitcherSuperSourceBox::GetCropped method

The **GetCropped** method returns the current cropped flag.

##### Syntax

```
HRESULT GetCropped (boolean* cropped);
```

##### Parameters

Name	Direction	Description
cropped	out	The current cropped flag.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The cropped parameter is invalid.

#### 6.2.4.12 IBMDSwitcherSuperSourceBox::SetCropped method

The **SetCropped** method sets the cropped flag.

##### Syntax

```
HRESULT SetCropped (boolean cropped);
```

##### Parameters

Name	Direction	Description
cropped	in	The desired cropped flag.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



#### 6.2.4.13 IBMDSwitcherSuperSourceBox::GetCropTop method

The **GetCropTop** method returns the current top crop value.

##### Syntax

```
HRESULT GetCropTop (double* top);
```

##### Parameters

Name	Direction	Description
top	out	The current top crop value.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The top parameter is invalid.

#### 6.2.4.14 IBMDSwitcherSuperSourceBox::SetCropTop method

The **SetCropTop** method sets the top crop value.

##### Syntax

```
HRESULT SetCropTop (double top);
```

##### Parameters

Name	Direction	Description
top	in	The desired top crop value.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.4.15 **IBMDSwitcherSuperSourceBox::GetCropBottom** method

The **GetCropBottom** method returns the current bottom crop value.

**Syntax**

```
HRESULT          GetCropBottom (double* bottom);
```

**Parameters**

Name	Direction	Description
bottom	out	The current bottom crop value.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The bottom parameter is invalid.

6.2.4.16 **IBMDSwitcherSuperSourceBox::SetCropBottom** method

The **SetCropBottom** method sets the bottom crop value.

**Syntax**

```
HRESULT          SetCropBottom (double bottom);
```

**Parameters**

Name	Direction	Description
bottom	in	The desired bottom crop value.

**Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.17 IBMDSwitcherSuperSourceBox::GetCropLeft method

The **GetCropLeft** method returns the current left crop value.

##### Syntax

```
HRESULT GetCropLeft (double* left);
```

##### Parameters

Name	Direction	Description
left	out	The current left crop value.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The left parameter is invalid.

#### 6.2.4.18 IBMDSwitcherSuperSourceBox::SetCropLeft method

The **SetCropLeft** method sets the left crop value.

##### Syntax

```
HRESULT SetCropLeft (double left);
```

##### Parameters

Name	Direction	Description
left	in	The desired left crop value.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.19 IBMDSwitcherSuperSourceBox::GetCropRight method

The **GetCropRight** method returns the current right crop value.

##### Syntax

```
HRESULT GetCropRight (double* right);
```

##### Parameters

Name	Direction	Description
right	out	The current right crop value.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The right parameter is invalid.

#### 6.2.4.20 IBMDSwitcherSuperSourceBox::SetCropRight method

The **SetCropRight** method sets the right crop value.

##### Syntax

```
HRESULT SetCropRight (double right);
```

##### Parameters

Name	Direction	Description
right	in	The desired right crop value.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.21 IBMDSwitcherSuperSourceBox::ResetCrop method

The **ResetCrop** method resets the crop to default values.

##### Syntax

```
HRESULT      ResetCrop (void);
```

##### Parameters

none.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 6.2.4.22 IBMDSwitcherSuperSourceBox::GetInputAvailabilityMask method

The **GetInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask for this supersource box. The input availability property (**bmdSwitcherInputPropertyIdInputAvailability**) of an **IBMDSwitcherInput** can be bitwise-ANDed with this mask value to determine whether an input is available for use as a source for this supersource box.

##### Syntax

**HRESULT**           GetInputAvailabilityMask (BMDSwitcherInputAvailability\* mask);

##### Parameters

Name	Direction	Description
mask	out	<b>BMDSwitcherInputAvailability</b> bit mask

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter

6.2.4.23 **IBMDSwitcherSuperSourceBox::AddCallback method**

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherSuperSourceBox** object. Pass an object implementing the **IBMDSwitcherSuperSourceBoxCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

**Syntax**

```
HRESULT          AddCallback (IBMDSwitcherSuperSourceBoxCallback* callback);
```

**Parameters**

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherSuperSourceBoxCallback</b> object interface.

**Return Values**

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 6.2.4.24 IBMDSwitcherSuperSourceBox::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherSuperSourceBoxCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherSuperSourceBoxCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



6.2.5

IBMDSwitcherSuperSourceBoxCallback Interface

The **IBMDSwitcherSuperSourceBoxCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherSuperSourceBox** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSuperSourceBox	IID_IBMDSwitcherSuperSourceBox	An <b>IBMDSwitcherSuperSourceBoxCallback</b> object interface is installed with <b>IBMDSwitcherSuperSourceBox::AddCallback</b> and removed with <b>IBMDSwitcherSuperSourceBox::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 6.2.5.1 **IBMDSwitcherSuperSourceBoxCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherSuperSourceBox** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

```
HRESULT Notify (BMDSwitcherSuperSourceBoxEventType eventType);
```

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherSuperSourceBoxEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7 Audio Mixing

Every switcher allows control over how audio is to be mixed into the program output, whether it is sourced from the media players, external audio-in or embedded with the video on an input.

### 7.1 Audio Mixing Data Types

#### 7.1.1 Audio Mixer Event Type

**BMDSwitcherAudioMixerEventType** enumerates the possible event types for the **IBMDSwitcherAudioMixerCallback** object interface.

<b>bmdSwitcherAudioMixerEventTypeProgramOutGainChanged</b>	The program out gain changed.
<b>bmdSwitcherAudioMixerEventTypeProgramOutBalanceChanged</b>	The program out balance changed.

#### 7.1.2 Audio Input Identifier

**BMDSwitcherAudioInputId**

**BMDSwitcherAudioInputId** is a signed 64 bit integer type and used as a unique identifier for each audio input.

#### 7.1.3 Audio Input Type

**BMDSwitcherAudioInputType** enumerates the possible input types for the **IBMDSwitcherAudioInput** object interface.

<b>bmdSwitcherAudioInputTypeEmbeddedWithVideo</b>	The audio is embedded into a switcher input.
<b>bmdSwitcherAudioInputTypeMediaPlayer</b>	The audio is from a media player.
<b>bmdSwitcherAudioInputTypeAudioIn</b>	The audio is from an external audio-in.

#### 7.1.4 Audio Mix Option

**BMDSwitcherAudioMixOption** enumerates the possible mix options for the **IBMDSwitcherAudioInput** object interface.

<b>bmdSwitcherAudioMixOptionOff</b>	The audio is not to be mixed into anything.
<b>bmdSwitcherAudioMixOptionOn</b>	The audio is always mixed into the output.
<b>bmdSwitcherAudioMixOptionAudioFollowVideo</b>	The audio is mixed into the output when its associated video is on air.

# SECTION 7 Audio Mixing

## 7.1.5 Audio Input Event Type

**BMDSwitcherAudioInputEventType** enumerates the possible event types for the **IBMDSwitcherAudioInputCallback** object interface.

<b>bmdSwitcherAudioInputEventTypeMixOptionChanged</b>	The mix option changed.
<b>bmdSwitcherAudioInputEventTypeGainChanged</b>	The gain changed.
<b>bmdSwitcherAudioInputEventTypeBalanceChanged</b>	The balance changed.
<b>bmdSwitcherAudioInputEventTypeCurrentExternalPortTypeChanged</b>	The audio input's external port type changed.
<b>bmdSwitcherAudioInputEventTypeIsMixedInChanged</b>	The is-mixed-in changed.

## 7.1.6 Audio Monitor Output Event Type

**BMDSwitcherAudioMonitorOutputEventType** enumerates the possible event types for the **IBMDSwitcherAudioMonitorOutputCallback** object interface.

<b>bmdSwitcherAudioMonitorOutputEventTypeMonitorEnableChanged</b>	The monitor enable flag changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeGainChanged</b>	The gain changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeMuteChanged</b>	The mute changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeSoloChanged</b>	The solo flag changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeSoloInputChanged</b>	The input that is soloed changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeDimChanged</b>	The dim flag changed.
<b>bmdSwitcherAudioMonitorOutputEventTypeDimLevelChanged</b>	The dim level changed.

## 7.1.7 Switcher Talkback Event Types

**BMDSwitcherTalkbackEventType** enumerates the possible event types for the **IBMDSwitcherTalkbackCallback** object interface.

<b>bmdSwitcherTalkbackEventTypeMuteSDIChanged</b>	The mute state of the talkback input SDI channels has changed.
---	--

7.2

Interface Reference

7.2.1

IBMDSwitcherAudioMixer Interface

The **IBMDSwitcherAudioMixer** object interface is the root object for all audio mixing control and feedback.

A reference to an **IBMDSwitcherAudioMixer** object interface may be obtained from an **IBMDSwitcher** object interface using the **QueryInterface** method. Pass **IID\_IBMDSwitcherAudioMixer** for the IID parameter.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	<b>IBMDSwitcher::QueryInterface</b> can return an <b>IBMDSwitcherAudioMixer</b> object interface.

Public Member Functions	
Method	Description
GetProgramOutGain	Get the current program out gain value.
SetProgramOutGain	Set the program out gain value.
GetProgramOutBalance	Get the current program out balance value.
SetProgramOutBalance	Set the program out balance value.
GetProgramOutFollowFadeToBlack	Get the current program out follow fade to black state.
SetProgramOutFollowFadeToBlack	Set the current program out follow fade to black state.
SetAllLevelNotificationsEnable	Opt-in to level notifications.
ResetProgramOutLevelNotificationPeaks	Reset program out peak level statistics to zero.
ResetAllLevelNotificationPeaks	Reset all switcher peak level statistics to zero.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.
CreateIterator	Create an iterator.

### 7.2.1.1 **IBMDSwitcherAudioMixer::GetProgramOutGain** method

The **GetProgramOutGain** method returns the current gain value.

#### Syntax

```
HRESULT GetProgramOutGain (double* gain);
```

#### Parameters

Name	Direction	Description
gain	out	The current gain value.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

### 7.2.1.2 **IBMDSwitcherAudioMixer::SetProgramOutGain** method

The **SetProgramOutGain** method sets the gain to apply to the program out.

#### Syntax

```
HRESULT SetProgramOutGain (double gain);
```

#### Parameters

Name	Direction	Description
gain	in	The desired gain value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.1.3 **IBMDSwitcherAudioMixer::GetProgramOutBalance** method

The **GetProgramOutBalance** method returns the current balance value.

#### **Syntax**

```
HRESULT GetProgramOutBalance (double* balance);
```

#### **Parameters**

Name	Direction	Description
balance	out	The current balance value.

#### **Return Values**

Value	Description
S_OK	Success.
E_POINTER	The balance parameter is invalid.

#### 7.2.1.4 IBMDSwitcherAudioMixer::GetProgramOutFollowFadeToBlack method

The **GetProgramOutFollowFadeToBlack** method returns the current follow fade to black state. When enabled the program out audio will fade in unity with a fade to black transition.

##### Syntax

**HRESULT**           GetProgramOutFollowFadeToBlack (boolean\* follow)

##### Parameters

Name	Direction	Description
follow	out	The current follow fade to black state.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The balance parameter is invalid.



#### 7.2.1.5 IBMDSwitcherAudioMixer::SetProgramOutFollowFadeToBlack method

The **SetProgramOutFollowFadeToBlack** method sets the current follow fade to black state. When enabled the program out audio will fade in unity with a fade to black transition.

##### Syntax

**HRESULT**            SetProgramOutFollowFadeToBlack (boolean follow)

##### Parameters

Name	Direction	Description
follow	in	The desired follow fade to black state.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 7.2.1.6 IBMDSwitcherAudioMixer::SetProgramOutBalance method

The **SetProgramOutBalance** method sets the balance to apply to the program out.

#### Syntax

```
HRESULT      SetProgramOutBalance (double balance);
```

#### Parameters

Name	Direction	Description
balance	in	The desired balance value.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 7.2.1.7 IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable method

The **SetAllLevelNotificationsEnable** method enables level statistics for the relevant mixer inputs and outputs. Receiving level notifications are an opt-in subscription, affecting the callbacks **IBMDSwitcherAudioMixerCallback::ProgramOutLevelNotification**, **IBMDSwitcherAudioInputCallback::LevelNotification** and **IBMDSwitcherAudioMonitorOutputCallback::LevelNotification**.

##### Syntax

**HRESULT** SetAllLevelNotificationsEnable (boolean enable);

##### Parameters

Name	Direction	Description
enable	in	Whether to enable notifications.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7.2.1.8 IBMDSwitcherAudioMixer::ResetProgramOutLevelNotificationPeaks method

The **ResetLevelNotificationPeaks** method resets's the switcher's program out peak level statistics to zero.

### Syntax

```
HRESULT      ResetProgramOutLevelNotificationPeaks (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7.2.1.9 IBMDSwitcherAudioMixer::ResetAllLevelNotificationPeaks method

The **ResetAllLevelNotificationPeaks** method resets peak statistics to zero for all mixer inputs and outputs.

### Syntax

```
HRESULT      ResetAllLevelNotificationPeaks (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.2.1.10 **IBMDSwitcherAudioMixer::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherAudioMixer** object. Pass an object implementing the **IBMDSwitcherAudioMixerCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

**Syntax**

```
HRESULT      AddCallback (IBMDSwitcherAudioMixerCallback* callback);
```

**Parameters**

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioMixerCallback</b> object interface.

**Return Values**

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

7.2.1.11 IBMDSwitcherAudioMixer::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback (IBMDSwitcherAudioMixerCallback* callback);
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioMixerCallback</b> object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

7.2.1.12 **IBMDSwitcherAudioMixer::CreateIterator method**

The **CreateIterator** method creates an iterator object interface for the specified interface ID, such as **IBMDSwitcherAudioInputIterator** and **IBMDSwitcherAudioMonitorOutputIterator**.

**Syntax**

```
HRESULT CreateIterator (REFIID iid, LPVOID* ppv);
```

**Parameters**

Name	Direction	Description
iid	in	Iterator Interface ID to create an iterator for.
ppv	in	Pointer to return interface object.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The ppv parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to create interface object.
E_NOINTERFACE	Interface was not found.

7.2.2 IBMDSwitcherAudioMixerCallback Interface

The **IBMDSwitcherAudioMixerCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherAudioMixer** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioMixer	IID_IBMDSwitcherAudioMixer	An <b>IBMDSwitcherAudioMixerCallback</b> object interface is installed with <b>IBMDSwitcherAudioMixer::AddCallback</b> and removed with <b>IBMDSwitcherAudioMixer::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
ProgramOutLevelNotification	Reports level statistics.



### 7.2.2.1 **IBMDSwitcherAudioMixerCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherAudioMixer** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **Notify** (BMDSwitcherAudioMixerEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherAudioMixerEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.2.2 IBMDSwitcherAudioMixerCallback::ProgramOutLevelNotification method

The **ProgramOutLevelNotification** method is called periodically to report the current dB levels and the last known peak levels. These peak levels can be reset using **IBMDSwitcherAudioMixer::ResetProgramOutLevelNotificationPeaks**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable**.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**           ProgramOutLevelNotification (double left, double right, peakRight);

#### Parameters

Name	Direction	Description
left	in	The current dB level of the left channel.
right	in	The current dB level of the right channel.
peakLeft	in	The highest encountered peak dB level of the left channel since the last reset.
peakRight	in	The highest encountered peak dB level of the right channel since the last reset.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.2.3

IBMDSwitcherAudioInputIterator Interface

The **IBMDSwitcherAudioInputIterator** is used to enumerate the available inputs for the audio mixer.

A reference to an **IBMDSwitcherAudioInputIterator** object interface may be obtained from an **IBMDSwitcherAudioMixer** object interface using the `CreateIterator` method. Pass **IID\_IBMDSwitcherAudioInputIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioMixer	IID_IBMDSwitcherAudioMixer	<b>IBMDSwitcherAudioMixer::CreateIterator</b> can return an <b>IBMDSwitcherAudioInputIterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherAudioInput</b> object interface.
GetById	Returns a pointer to an <b>IBMDSwitcherAudioInput</b> object interface, given its <b>BMDSwitcherAudioInputId</b> .

### 7.2.3.1 **IBMDSwitcherAudioInputIterator::Next** method

The **Next** method returns the next available **IBMDSwitcherAudioInput** object interface.

#### Syntax

```
HRESULT         Next (IBMDSwitcherAudioInput** audioInput);
```

#### Parameters

Name	Direction	Description
audioInput	out	<b>IBMDSwitcherAudioInput</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherAudioInput</b> objects available.
E_POINTER	The audioInput parameter is invalid.

### 7.2.3.2 **IBMDSwitcherAudioInputIterator::GetById** method

The **GetById** method returns a pointer to an **IBMDSwitcherAudioInput** object interface, given its **BMDSwitcherAudioInputId**.

#### Syntax

```
HRESULT          GetById (BMDSwitcherAudioInputId audioInputId, IBMDSwitcherAudioInput** audioInput);
```

#### Parameters

Name	Direction	Description
audioInputId	in	<b>BMDSwitcherAudioInputId</b> identifier.
audioInput	out	<b>IBMDSwitcherAudioInput</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioInputId is not a valid identifier.
E_POINTER	The audioInput parameter is invalid.

7.2.4 IBMDSwitcherAudioInput Interface

The **IBMDSwitcherAudioInput** object interface is used for manipulating the settings of an audio input.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioInputIterator	IID_ IBMDSwitcherAudioInputIterator	An <b>IBMDSwitcherAudioInput</b> object interface will be returned after a successful call to <b>IBMDSwitcherAudioInputIterator::Next</b> method.

Public Member Functions	
Method	Description
GetType	Get the audio input type.
GetMixOption	Get the current mix option.
SetMixOption	Set the mix option.
GetGain	Get the current gain value.
SetGain	Set the gain value.
GetBalance	Get the current balance value.
SetBalance	Set the balance value.
IsMixedIn	Get the current is-mixed-in flag.
GetAudioInputId	Get the identifier of the audio input.
ResetLevelNotificationPeaks	Reset peak level statistics to zero.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

#### 7.2.4.1 IBMDSwitcherAudioInput::GetType method

The **GetType** method returns the type of the audio input.

##### Syntax

```
HRESULT      GetType (BMDSwitcherAudioInputType* type);
```

##### Parameters

Name	Direction	Description
type	out	The audio input type.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The type parameter is invalid.

#### 7.2.4.2 **IBMDSwitcherAudioInput::GetCurrentExternalPortType** method

The **GetCurrentExternalPortType** method gets the current physical external port type of the audio input. This may change if the physical input is switchable, generating the event **bmdSwitcherAudioInputEventTypeCurrentExternalPortTypeChanged**.

##### Syntax

**HRESULT**            **GetCurrentExternalPortType** (BMDSwitcherExternalPortType\* type);

##### Parameters

Name	Direction	Description
type	out	The current external port type.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The type parameter is invalid.



### 7.2.4.3 **IBMDSwitcherAudioInput::GetMixOption** method

The **GetMixOption** method returns the mix option of the audio input.

#### **Syntax**

```
HRESULT GetMixOption (BMDSwitcherAudioMixOption* mixOption);
```

#### **Parameters**

Name	Direction	Description
mixOption	out	The audio input mix option.

#### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The mixOption parameter is invalid.

#### 7.2.4.4 **IBMDSwitcherAudioInput::SetMixOption** method

The **SetMixOption** method sets the mix option of the audio input.

##### Syntax

```
HRESULT          SetMixOption (BMDSwitcherAudioMixOption mixOption);
```

##### Parameters

Name	Direction	Description
mixOption	in	The audio input mix option.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The mixOption parameter is invalid.

## 7.2.4.5 IBMDSwitcherAudioInput::GetGain method

The **GetGain** method returns the gain currently applied to the audio input.

### Syntax

```
HRESULT GetGain (double* gain);
```

### Parameters

Name	Direction	Description
gain	out	The gain currently applied to the audio input.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

## 7.2.4.6 IBMDSwitcherAudioInput::SetGain method

The **SetGain** method sets the gain to apply to the audio input.

### Syntax

```
HRESULT SetGain (double gain);
```

### Parameters

Name	Direction	Description
gain	in	The gain to apply to the audio input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 7.2.4.7 IBMDSwitcherAudioInput::GetBalance method

The **GetBalance** method returns the current balance.

##### Syntax

```
HRESULT          GetBalance (double* balance);
```

##### Parameters

Name	Direction	Description
balance	out	The current balance.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The balance parameter is invalid.

#### 7.2.4.8 IBMDSwitcherAudioInput::SetBalance method

The **SetBalance** method sets the balance to apply to the audio input.

##### Syntax

```
HRESULT      SetBalance (double balance);
```

##### Parameters

Name	Direction	Description
balance	in	The balance to apply.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The balance parameter is invalid.

#### 7.2.4.9 IBMDSwitcherAudioInput::IsMixedIn method

The **IsMixedIn** method indicates whether the audio input is currently being mixed into the program out.

##### Syntax

```
HRESULT      IsMixedIn (boolean* mixedIn);
```

##### Parameters

Name	Direction	Description
mixedIn	out	The current mixed-in flag.

##### Return Values

Value	Description
S_OK	Success.
E_POINTER	The mixedIn parameter is invalid.

## 7.2.4.10 IBMDSwitcherAudioInput::GetAudioInputId method

The **GetAudioInputId** method gets the **BMDSwitcherAudioInputId** of the audio input.

### Syntax

```
HRESULT GetAudioInputId (BMDSwitcherAudioInputId* audioInputId);
```

### Parameters

Name	Direction	Description
audioInputId	out	The <b>BMDSwitcherAudioInputId</b> of the audio input.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7.2.4.11 IBMDSwitcherAudioInput::ResetLevelNotificationPeaks method

The **ResetLevelNotificationPeaks** method resets the switcher's input peak level statistics to zero.

### Syntax

```
HRESULT ResetLevelNotificationPeaks (void);
```

### Parameters

none.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 7.2.4.12 IBMDSwitcherAudioInput::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherAudioInput** object. Pass an object implementing the **IBMDSwitcherAudioInputCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

##### Syntax

**HRESULT**           AddCallback (IBMDSwitcherAudioInputCallback\* callback);

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioInputCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.



#### 7.2.4.13 IBMDSwitcherAudioInput::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherAudioInputCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioInputCallback</b> object interface.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

7.2.5

IBMDSwitcherAudioInputCallback Interface

The **IBMDSwitcherAudioInputCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherAudioInput** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioInput	IID_IBMDSwitcherAudioInput	An <b>IBMDSwitcherAudioInputCallback</b> object interface is installed with <b>IBMDSwitcherAudioInput::AddCallback</b> and removed with <b>IBMDSwitcherAudioInput::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
LevelNotification	Reports level statistics.

### 7.2.5.1 **IBMDSwitcherAudioInputCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherAudioInput** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **Notify** (BMDSwitcherAudioInputEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherAudioInputEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.5.2 **IBMDSwitcherAudioInputCallback::LevelNotification** method

The **LevelNotification** method is called periodically to report the current dB levels and the last known peak levels. These peak levels can be reset using **IBMDSwitcherAudioInput::ResetLevelNotificationPeaks**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable**.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **LevelNotification** (double left, double right, double peakLeft, double peakRight);

#### Parameters

Name	Direction	Description
left	in	The current dB level of the left channel.
right	in	The current dB level of the right channel.
peakLeft	in	The highest encountered peak dB level of the left channel since the last reset.
peakRight	in	The highest encountered peak dB level of the right channel since the last reset.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.2.6

IBMDSwitcherAudioMonitorOutputIterator Interface

The **IBMDSwitcherAudioMonitorOutputIterator** is used to enumerate the available monitor outputs for the audio mixer.

A reference to an **IBMDSwitcherAudioMonitorOutputIterator** object interface may be obtained from an **IBMDSwitcherAudioMixer** object interface using the **CreateIterator** method. Pass **IID\_IBMDSwitcherAudioMonitorOutputIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioMixer	IID_IBMDSwitcherAudioMixer	<b>IBMDSwitcherAudioMixer::CreateIterator</b> can return an <b>IBMDSwitcherAudioMonitorOutputIterator</b> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <b>IBMDSwitcherAudioMonitorOutputIterator</b> object interface.

7.2.6.1 IBMDSwitcherAudioMonitorOutputIterator::Next method

The **Next** method returns the next available **IBMDSwitcherAudioMonitorOutput** object interface.

Syntax

```
HRESULT      Next (IBMDSwitcherAudioMonitorOutput** audioMonitorOutput);
```

Parameters

Name	Direction	Description
audioMonitorOutput	out	<b>IBMDSwitcherAudioMonitorOutput</b> object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <b>IBMDSwitcherAudioMonitorOutput</b> objects available.
E_POINTER	The audioMonitorOutput parameter is invalid.

7.2.7 IBMDSwitcherAudioMonitorOutput Interface

The **IBMDSwitcherAudioMonitorOutput** object interface is used for manipulating parameters specific to audio monitor outputs.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioMonitorOutput Iterator	IID_ IBMDSwitcherAudioMonitorOutput	An <b>IBMDSwitcherAudioMonitorOutput</b> interface can be obtained with <b>IBMDSwitcherAudioMonitorOutputIterator::Next</b> .

Public Member Functions	
Method	Description
GetMonitorEnable	Get the current monitor-enable flag.
SetMonitorEnable	Set the monitor-enable flag.
GetMute	Get the current mute flag.
SetMute	Set the mute flag.
GetGain	Get the current gain value.
SetGain	Set the gain value.
GetSolo	Get the current solo flag.
SetSolo	Set the solo flag.
GetSoloInput	Get the current soloed input.
SetSoloInput	Set the soloed input.
GetDim	Get the current dim flag.
SetDim	Set the dim flag.
GetDimLevel	Get the current dim level.
SetDimLevel	Set the dim level.

# SECTION 7 Audio Mixing

Public Member Functions	
Method	Description
ResetLevelNotificationPeaks	Reset peak level statistics to zero.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.



7.2.7.1

IBMDSwitcherAudioMonitorOutput::GetMonitorEnable method

The **GetMonitorEnable** method returns the current monitor enable flag.

**Syntax**

```
HRESULT      GetMonitorEnable (boolean* enable);
```

**Parameters**

Name	Direction	Description
enable	out	The current monitor enable flag.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The enable parameter is invalid.

### 7.2.7.2 IBMDSwitcherAudioMonitorOutput::SetMonitorEnable method

The **SetMonitorEnable** method sets the monitor enable flag.  
 This output acts as a monitor when the flag is set, otherwise it mirrors the content of program out.

#### Syntax

```
HRESULT      SetMonitorEnable (boolean enable);
```

#### Parameters

Name	Direction	Description
enable	in	The current monitor enable flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.7.3 IBMDSwitcherAudioMonitorOutput::GetMute method

The **GetMute** method returns the current mute flag.

#### Syntax

```
HRESULT GetMute (boolean* mute);
```

#### Parameters

Name	Direction	Description
mute	out	The current mute flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The mute parameter is invalid.

### 7.2.7.4 IBMDSwitcherAudioMonitorOutput::SetMute method

The **SetMute** method sets the mute flag.

#### Syntax

```
HRESULT SetMute (boolean mute);
```

#### Parameters

Name	Direction	Description
mute	in	The desired mute flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7.2.7.5 IBMDSwitcherAudioMonitorOutput::GetGain method

The **GetGain** method returns the current gain value.

### Syntax

```
HRESULT GetGain (double* gain);
```

### Parameters

Name	Direction	Description
gain	out	The current gain value.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

## 7.2.7.6 IBMDSwitcherAudioMonitorOutput::SetGain method

The **SetGain** method sets the gain to apply to the audio monitor output.

### Syntax

```
HRESULT SetGain (double gain);
```

### Parameters

Name	Direction	Description
gain	in	The desired gain value.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 7.2.7.7 IBMDSwitcherAudioMonitorOutput::GetSolo method

The **GetSolo** method returns the current solo flag.

### Syntax

```
HRESULT GetSolo (boolean* solo);
```

### Parameters

Name	Direction	Description
solo	out	The current solo flag.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The solo parameter is invalid.

## 7.2.7.8 IBMDSwitcherAudioMonitorOutput::SetSolo method

The **SetSolo** method sets the solo flag.

### Syntax

```
HRESULT SetSolo (boolean solo);
```

### Parameters

Name	Direction	Description
solo	in	The desired solo flag.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.2.7.9

IBMDSwitcherAudioMonitorOutput::GetSoloInput method

The **GetSoloInput** method returns which audio input is selected for soloing in the monitor output.

**Syntax**

**HRESULT**            GetSoloInput (BMDSwitcherAudioInputId\* audioInput);

**Parameters**

Name	Direction	Description
audioInput	out	The audio input for soloing.

**Return Values**

Value	Description
S_OK	Success.
E_POINTER	The audioInput parameter is invalid.

#### 7.2.7.10 IBMDSwitcherAudioMonitorOutput::SetSoloInput method

The **SetSoloInput** method selects which audio input is soloed in the monitor output.

##### Syntax

```
HRESULT      SetSoloInput (BMDSwitcherAudioInputId audioInput);
```

##### Parameters

Name	Direction	Description
audioInput	in	The audio input for soloing.

##### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The audioInput parameter is invalid.

### 7.2.7.11 IBMDSwitcherAudioMonitorOutput::GetDim method

The **GetDim** method returns the current dim flag.

#### Syntax

```
HRESULT      GetDim (boolean* dim);
```

#### Parameters

Name	Direction	Description
dim	out	The current dim flag.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The dim parameter is invalid.

### 7.2.7.12 IBMDSwitcherAudioMonitorOutput::SetDim method

The **SetDim** method sets the dim flag.

#### Syntax

```
HRESULT      SetDim (boolean dim);
```

#### Parameters

Name	Direction	Description
dim	in	The desired dim flag.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.



## 7.2.7.13 IBMDSwitcherAudioMonitorOutput::GetDimLevel method

The **GetDimLevel** method returns the current dim level in dB.

### Syntax

```
HRESULT GetDimLevel (double* gain);
```

### Parameters

Name	Direction	Description
gain	out	The current dim level.

### Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.

## 7.2.7.14 IBMDSwitcherAudioMonitorOutput::SetDimLevel method

The **SetDimLevel** method sets the dim level in dB.

### Syntax

```
HRESULT SetDimLevel (double gain);
```

### Parameters

Name	Direction	Description
dim	in	The desired dim level.

### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

#### 7.2.7.15 **IBMDSwitcherAudioMonitorOutput::ResetLevelNotificationPeaks** method

The **ResetLevelNotificationPeaks** method resets the switcher's output peak level statistics to zero.

##### **Syntax**

```
HRESULT ResetLevelNotificationPeaks (void);
```

##### **Parameters**

none.

##### **Return Values**

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.7.16 IBMDSwitcherAudioMonitorOutput::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherAudioMonitorOutput** object. Pass an object implementing the **IBMDSwitcherAudioMonitorOutputCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

**HRESULT**           AddCallback (IBMDSwitcherAudioMonitorOutputCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioMonitorOutputCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

### 7.2.7.17 **IBMDSwitcherAudioMonitorOutput::RemoveCallback** method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

**HRESULT** RemoveCallback (IBMDSwitcherAudioMonitorOutputCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherAudioMonitorOutputCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

7.2.8

IBMDSwitcherAudioMonitorOutputCallback Interface

The **IBMDSwitcherAudioMonitorOutputCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherAudioMonitorOutput** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioMonitorOutput	IID_ IBMDSwitcherAudioMonitorOutput	An <b>IBMDSwitcherAudioMonitorOutputCallback</b> object interface is installed with <b>IBMDSwitcherAudioMonitorOutput::AddCallback</b> and removed with <b>IBMDSwitcherAudioMonitorOutput::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
LevelNotification	Reports level statistics.

### 7.2.8.1 IBMDSwitcherAudioMonitorOutputCallback::Notify method

The **Notify** method is called when **IBMDSwitcherAudioMonitorOutput** events occur, such as property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            Notify (BMDSwitcherAudioMonitorOutputEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherAudioMonitorOutputEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.8.2 **IBMDSwitcherAudioMonitorOutputCallback::LevelNotification** method

The **LevelNotification** method is called periodically to report the current dB levels and the last known peak levels. These peak levels can be reset using **IBMDSwitcherAudioMonitorOutput::ResetLevelNotificationPeaks**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable**.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **LevelNotification** (double left, double right, double peakLeft, double peakRight);

#### Parameters

Name	Direction	Description
left	in	The current dB level of the left channel.
right	in	The current dB level of the right channel.
peakLeft	in	The highest encountered peak dB level of the left channel since the last reset.
peakRight	in	The highest encountered peak dB level of the right channel since the last reset.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.2.9 IBMDSwitcherTalkback Interface

The **IBMDSwitcherTalkback** object interface is used for managing functionality relating to the talkback features on the switcher.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An <b>IBMDSwitcherTalkback</b> interface can be obtained with <b>IBMDSwitcher::QueryInterface</b> .

Public Member Functions	
Method	Description
GetMuteSDI	Query the mute state of the talkback input SDI channels.
SetMuteSDI	Set the mute state of the talkback input SDI channels.
AddCallback	Add a talkback callback.
RemoveCallback	Remove a talkback callback.



### 7.2.9.1 IBMDSwitcherTalkback::GetMuteSDI method

The **GetMuteSDI** method returns the mute state of the audio on the dedicated talkback input SDI channels.

#### Syntax

```
HRESULT GetMuteSDI (boolean* muteSDI);
```

#### Parameter

Name	Direction	Description
muteSDI	out	The mute state of the talkback input SDI channels.

#### Return Values

Value	Description
S_OK	Success.
E_POINTER	The muteSDI parameter is not a valid pointer.

### 7.2.9.2 IBMDSwitcherTalkback::SetMuteSDI method

The **SetMuteSDI** method sets the mute state of the audio on the dedicated talkback input SDI channels.

#### Syntax

```
HRESULT      SetMuteSDI (boolean muteSDI);
```

#### Parameters

Name	Direction	Description
muteSDI	out	The mute state of the talkback input SDI channels.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

### 7.2.9.3 IBMDSwitcherTalkback::AddCallback method

The **AddCallback** method configures a callback to be called when a switcher talkback property changes, such as the talkback SDI mute state.

Adding a new callback will not affect previously added callbacks.

Callbacks will be called on a separate thread and in the order of their addition by **AddCallback**.

#### Syntax

```
HRESULT          AddCallback (IBMDSwitcherTalkbackCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherTalkbackCallback</b> interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

#### 7.2.9.4 IBMDSwitcherTalkback::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

##### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherTalkbackCallback* callback);
```

##### Parameters

Name	Direction	Description
callback	in	Callback object to remove.

##### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

7.2.10 IBMDSwitcherTalkbackCallback Interface

The **IBMDSwitcherTalkbackCallback** object interface is a callback class which is called when a switcher talkback event occurs, such as a change in the talkback input SDI mute state.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherTalkback	IID_IBMDSwitcherTalkback	An <b>IBMDSwitcherTalkbackCallback</b> interface may be installed with <b>IBMDSwitcherTalkback::AddCallback</b> .

Public Member Functions	
Method	Description
Notify	A Switcher talkback event occurred, such as a change in the talkback input SDI mute state.

### 7.2.10.1 **IBMDSwitcherSerialPortCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherTalkback** events occur.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads.

Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### Syntax

**HRESULT**            **Notify** (BMDSwitcherTalkbackEventType eventType);

#### Parameters

Name	Direction	Description
eventType	in	A <b>BMDSwitcherTalkbackEventType</b> that describes the type of event that has occurred.

#### Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

## 8 Camera Control

The Switcher Camera Control API provides programmatic access to supported Blackmagic Design cameras via ATEM switchers. Devices must support the Blackmagic Video Device Embedded Control Protocol. Please refer to the Developer Information section of the ATEM Switchers Operation Manual for further information about the protocol.

### 8.1 Camera Control Data Types

#### 8.1.1 Switcher Camera Control Event Type

**BMDSwitcherCameraControlEventType** enumerates the possible event types for the **IBMDSwitcherCameraControlCallback** object interface.

<b>bmdSwitcherCameraControlEventTypePeriodicFlushIntervalChanged</b>	The periodic flush interval has changed.
<b>bmdSwitcherCameraControlEventTypeParameterValueChanged</b>	The parameter value has changed.
<b>bmdSwitcherCameraControlEventTypeParameterPeriodicFlushEnabledChanged</b>	The parameter period flush enabled state has changed.

#### 8.1.2 Switcher Camera Control Parameter Type

**BMDSwitcherCameraControlParameterType** enumerates the possible parameter types for the **IBMDSwitcherCameraControl** object interface.

<b>bmdSwitcherCameraControlParameterTypeVoidBool</b>	Boolean (1 or more values) or Void (0 values).
<b>bmdSwitcherCameraControlParameterTypeSigned8Bit</b>	Signed 8-bit type. This type is the same as byte.
<b>bmdSwitcherCameraControlParameterTypeSigned16Bit</b>	Signed 16-bit type
<b>bmdSwitcherCameraControlParameterTypeSigned32Bit</b>	Signed 32-bit type
<b>bmdSwitcherCameraControlParameterTypeSigned64Bit</b>	Signed 64-bit type
<b>bmdSwitcherCameraControlParameterTypeFixedPoint16Bit</b>	Binary fixed point signed number. 5 bits integer and 11 bits fractional.

8.2

Interface Reference

8.2.1

Switcher Camera Control Parameter Iterator

The **IBMDSwitcherCameraControlParameterIterator** is used to iterate control parameters that have been previously set. A reference to an **IBMDSwitcherCameraControlParameterIterator** object interface may be obtained from an **IBMDSwitcherCameraControl** object interface using the **CreateIterator** method.

The **IBMDSwitcherCameraControlParameterIterator** interface is used to iterate through control parameters.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherCameraControl	IID_IBMDSwitcherCameraControl	<b>IBMDSwitcherCameraControl::CreateIterator</b> returns an <b>IBMDSwitcherCameraControlParameterIterator</b> object interface when the IID_IBMDSwitcherCameraControlParameterIterator IID is specified.

Public Member Functions	
Method	Description
Next	Returns the next control parameter.



### 8.2.1.1 IBMDSwitcherCameraControlParameterIterator::Next method

The **Next** method returns the next control parameter.

#### Syntax

**HRESULT**            Next(uint32\_t\* destinationDevice, uint32\_t\* category, uint32\_t\* parameter)

#### Parameters

Name	Direction	Description
destinationDevice	out	The destination device address.
category	out	The configuration category number.
parameter	out	The configuration parameter number.

#### Return Values

Value	Description
S_FALSE	No more control parameters are available.
S_OK	Success.
E_POINTER	One or more of the parameters is NULL.

### 8.2.2 IBMDSwitcherCameraControlCallback Interface

The **IBMDSwitcherCameraControlCallback** object interface is a callback class which is called when a camera control event occurs.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherCameraControl	IID_IBMDSwitcherCameraControl	An <b>IBMDSwitcherCameraControlCallback</b> object interface may be installed with <b>IBMDSwitcherCameraControl::AddCallback</b> .

Public Member Functions	
Method	Description
Notify	Called when a camera control event occurs.

### 8.2.2.1 IBMDSwitcherCameraControlCallback::Notify method

The **Notify** method is called when **IBMDSwitcherCameraControl** events occur. This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher. To determine the type and count of the parameter values call **IBMDSwitcherCameraControl::GetParameterInfo**.

#### Syntax

```
HRESULT Notify (BMDSwitcherCameraControlEventType eventType, uint32_t destinationDevice,
                uint32_t category, uint32_t parameter)
```

#### Parameters

Name	Direction	Description
eventType	in	The BMDSwitcherCameraControlEventType that describes the type of event that has occurred.
destinationDevice	in	The destination device address.
category	in	The configuration category number available on the device.
parameter	in	The configuration parameter number available on the device.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	A parameter is invalid.

### 8.2.3 IBMDSwitcherCameraControl Interface

The **IBMDSwitcherCameraControl** object interface is used for controlling compatible Blackmagic Design cameras.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_ IBMDSwitcher	An <b>IBMDSwitcherCameraControl</b> object interface can be obtained with <b>IBMDSwitcher::QueryInterface</b> .

Public Member Functions	
Method	Description
CreateIterator	Create an iterator.
GetPeriodicFlushInterval	Get the periodic flush interval for parameters to be sent over SDI.
SetPeriodicFlushInterval	Set the periodic flush interval for parameters to be sent over SDI.
GetParameterInfo	Get the type and count of a parameter.
GetParameterPeriodicFlushEnabled	Get the status of flush enable.
SetFlags	Sets the current flag values.
ToggleFlags	Toggles the current flag values.
GetFlags	Gets the current flag values.
SetBytes	Sets the current signed 8-bit values.
OffsetBytes	Apply signed offsets to the signed 8-bit values.
GetBytes	Get the parameter values consisting of bytes.
SetInt16s	Sets the current signed 16-bit integer values.
OffsetInt16s	Apply a signed offset to the signed 16-bit integer values.
GetInt16s	Get the current signed 16-bit integer values.
SetInt32s	Sets the current signed 32-bit integer values.
OffsetInt32s	Apply a signed offset to the signed 32-bit integer values.

## SECTION 8 Camera Control

Public Member Functions	
Method	Description
GetInt32s	Get the current signed 32-bit integer values.
SetInt64s	Sets the current signed 64-bit integer values.
OffsetInt64s	Apply a signed offset to the signed 64-bit integer values.
GetInt64s	Get the current signed 64-bit integer values.
OffsetFloats	Apply a signed floating point offset to the fixed point values.
SetFloats	Sets the current fixed point values.
GetFloats	Get the current fixed point values.
AddCallback	Add a callback to receive camera control event notifications.
RemoveCallback	Remove a callback.

### 8.2.3.1 IBMDSwitcherCameraControl::CreateIterator method

The **CreateIterator** method creates an iterator object interface for the specified interface ID.

#### Syntax

**HRESULT** CreateIterator(REFIID iid, LPVOID\* ppv)

#### Parameters

Name	Direction	Description
iid	in	Iterator Interface ID to create an iterator for.
ppv	in	Pointer to return interface object.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_OUTOFMEMORY	Unable to allocate memory.
E_POINTER	The ppv parameter is NULL.
E_NOINTERFACE	Unable to locate interface matching iid parameter.

### 8.2.3.2 IBMDSwitcherCameraControl::GetPeriodicFlushInterval method

The **GetPeriodicFlushInterval** method returns the periodic interval set on the switcher. The flush interval is the period where the nominated parameters are sent periodically from the switcher over the SDI connection to the cameras.

#### Syntax

```
HRESULT GetPeriodicFlushInterval (uint32_t *intervalMs);
```

#### Parameters

Name	Direction	Description
intervalMs	out	The periodic flush interval in milliseconds.

#### Return Values

Value	Description
E_POINTER	The intervalMs parameter is NULL.
S_OK	Success.

### 8.2.3.3 IBMDSwitcherCameraControl::SetPeriodicFlushInterval method

The **SetPeriodicFlushInterval** method sets the periodic flush interval on the switcher.

The flush interval is the period where the parameters are sent periodically from the switcher over the SDI connection to the cameras.

#### Syntax

```
HRESULT SetPeriodicFlushInterval (uint32_t intervalMs);
```

#### Parameters

Name	Direction	Description
intervalMs	in	The refresh interval in milliseconds.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.



#### 8.2.3.4 IBMDSwitcherCameraControl::GetParameterInfo method

The **GetParameterInfo** method obtains the type of value (**BMDSwitcherCameraControlParameterType**) and the number of values for a given parameter.

##### Syntax

```
HRESULT GetParameterInfo (uint32_t destinationDevice, uint32_t category, uint32_t parameter,
                           BMDSwitcherCameraControlParameterType* type, uint32_t* count);
```

##### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The configuration category number.
parameter	in	The parameter number.
type	out	The parameter type.
count	out	Number of values.

##### Return Values

Value	Description
S_OK	Success.
E_UNEXPECTED	An unexpected type error has occurred.
E_INVALIDARG	Unable to find parameter information for the given arguments.
E_POINTER	The type or count parameter is NULL.

## 8.2.3.5 IBMDSwitcherCameraControl::GetParameterPeriodicFlushEnabled method

The **GetParameterPeriodicFlushEnabled** method returns the periodic flush enabled status for the parameter values. When enabled, the parameter values will be flushed periodically from the switcher over the SDI connection to the cameras. The flush interval can be changed by calling **SetPeriodicFlushInterval**.

### Syntax

**HRESULT** GetParameterPeriodicFlushEnabled (uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, boolean\* nabled)

### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The configuration category number.
parameter	in	The parameter number.
enabled	out	The periodic flush enabled status.

### Return Values

Value	Description
E_POINTER	The enabled parameter is NULL.
S_OK	Success.
E_INVALIDARG	Unable to find an entry for the given arguments.

### 8.2.3.6 IBMDSwitcherCameraControl::SetParameterPeriodicFlushEnabled method

The **SetParameterPeriodicFlushEnabled** method sets the periodic flush enabled parameters. The parameters will not be flushed unless this option has been enabled. The flush interval is the period where the parameters are sent periodically from the switcher over the SDI connection to the cameras.

#### Syntax

```
HRESULT SetParameterPeriodicFlushEnabled (uint32_t destinationDevice, uint32_t category,
uint32_t parameter, boolean enabled)
```

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
enabled	in	Enable periodic flush.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 8.2.3.7 IBMDSwitcherCameraControl::SetFlags method

The **SetFlags** method sends flags to the cameras over SDI.

#### Syntax

```
HRESULT SetFlags (uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const boolean* values)
```

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	in	The number of parameter elements.
values	in	The parameter values.

#### Return Values

Value	Description
E_FAIL	Failure .
S_OK	Success.
E_POINTER	The values parameter is NULL and the count is non-zero.
E_OUTOFMEMORY	There is insufficient memory to complete operation.

## 8.2.3.8 IBMDSwitcherCameraControl::ToggleFlags method

The **ToggleFlags** method will toggle the flags and then send the flags from the switcher to the cameras over SDI. If the parameter value is true, then the flag will be toggled otherwise it will remain the same.

### Syntax

```
HRESULT ToggleFlags(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const boolean* values)
```

### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of parameter elements.
<code>values</code>	in	The parameter values.

### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values parameter is NULL and the count is non-zero.
<code>E_OUTOFMEMORY</code>	There is insufficient memory to complete operation.

### 8.2.3.9 IBMDSwitcherCameraControl::GetFlags method

The **GetFlags** method returns the last flags sent to the cameras over SDI.

#### Syntax

```
HRESULT          GetFlags(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
                          uint32_t* count, boolean* values)
```

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	out	The element count.
values	out	The parameter values.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_INVALIDARG	Flags do not exist for the given arguments.
E_POINTER	The count or values parameter is NULL.

## 8.2.3.10 IBMDSwitcherCameraControl::SetBytes method

The SetBytes method sends 8-bit values to the cameras over SDI.

### Syntax

```
HRESULT SetBytes(uint32_t destinationDevice, uint32_t category, uint32_t parameter, uint32_t count, const int8_t* bytes)
```

### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of elements.
<code>bytes</code>	in	The parameter values.

### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The bytes argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

### 8.2.3.11 IBMDSwitcherCameraControl::OffsetBytes method

The OffsetBytes method will add signed offsets to the current 8-bit values and then send the offset values to the cameras over SDI.

#### Syntax

**HRESULT**            OffsetBytes(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t count, const int8\_t\* bytes)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	in	The number of elements.
bytes	in	The parameter values.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The bytes argument is NULL.
E_OUTOFMEMORY	There was insufficient memory to complete the operation.



### 8.2.3.12 IBMDSwitcherCameraControl::GetBytes method

The GetBytes method returns the last signed bytes sent to the cameras over SDI.

#### Syntax

**HRESULT** GetBytes(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t\* count, int8\_t\* bytes)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	out	The element count.
bytes	out	The parameter values.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	Bytes do not exist for the given arguments.
E_POINTER	The count or byte parameter is NULL.

### 8.2.3.13 IBMDSwitcherCameraControl::SetInt16s method

The **SetInt16s** method sends signed 16-bit values from the switcher to the cameras over SDI.

#### Syntax

```
HRESULT SetInt16s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int16_t* values)
```

#### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of elements.
<code>values</code>	in	The parameter values.

#### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

#### 8.2.3.14 IBMDSwitcherCameraControl::OffsetInt16s method

The **OffsetInt16s** method will add signed offsets to the current 16-bit values and then send the offset values to the cameras over SDI.

##### Syntax

```
HRESULT OffsetInt16s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int16_t* values)
```

##### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of elements.
<code>values</code>	in	The parameter values.

##### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

### 8.2.3.15 IBMDSwitcherCameraControl::GetInt16s method

The **GetInt16s** method returns the last 16-bit signed integers sent to the cameras over SDI.

#### Syntax

**HRESULT**            GetInt16s(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t\* count, int16\_t\* values)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	out	The number of parameter elements.
values	out	The parameter values.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	Int16s do not exist for the given arguments.
E_POINTER	The count or values argument is NULL.

## 8.2.3.16 IBMDSwitcherCameraControl::SetInt32s method

The **SetInt32s** method sends signed 32-bit values to the cameras over SDI.

### Syntax

```
HRESULT SetInt32s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int32_t* values)
```

### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of parameter elements.
<code>values</code>	in	The parameter values.

### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

### 8.2.3.17 IBMDSwitcherCameraControl::OffsetInt32s method

The **OffsetInt32s** method will add signed offsets to the current 32-bit values and then send the offset values to the cameras over SDI.

#### Syntax

```
HRESULT OffsetInt32s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int32_t* values)
```

#### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of elements.
<code>values</code>	in	The parameter values.

#### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

### 8.2.3.18 IBMDSwitcherCameraControl::GetInt32s method

The **GetInt32s** method returns the last 32-bit signed integers sent to the cameras over SDI.

#### Syntax

**HRESULT**            GetInt32s(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t\* count, int32\_t\* values)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	out	The number of elements.
values	out	The parameter values.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	Int32s do not exist for the given arguments.
E_POINTER	The count or values argument is NULL.

### 8.2.3.19 IBMDSwitcherCameraControl::SetInt64s method

The **SetInt64s** method sends signed 64-bit values from the switcher to the cameras over SDI.

#### Syntax

```
HRESULT SetInt64s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int64_t* values)
```

#### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The number of parameter elements.
<code>values</code>	in	The parameter values.

#### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.



### 8.2.3.20 IBMDSwitcherCameraControl::OffsetInt64s method

The **OffsetInt64s** method will add signed offsets to the current 64-bit values and then send the offset values to the cameras over SDI.

#### Syntax

```
HRESULT OffsetInt64s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t count, const int64_t* values)
```

#### Parameters

Name	Direction	Description
<b>destinationDevice</b>	in	The destination device address.
<b>category</b>	in	The category.
<b>parameter</b>	in	The parameter.
<b>count</b>	in	The number of elements.
<b>values</b>	in	The parameter values.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The values argument is NULL.
E_OUTOFMEMORY	There was insufficient memory to complete the operation.

### 8.2.3.21 IBMDSwitcherCameraControl::GetInt64s method

The **GetInt64s** method returns the last 64-bit signed integers sent to the cameras over SDI.

#### Syntax

```
HRESULT GetInt64s(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
uint32_t* count, int64_t* values)
```

#### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	out	The element count.
<code>values</code>	out	The parameter values.

#### Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_INVALIDARG</code>	Int64s do not exist for the given arguments.
<code>E_POINTER</code>	The count or values argument is NULL.

### 8.2.3.22 IBMDSwitcherCameraControl::OffsetFloats method

The **OffsetFloats** method will add signed offsets to the current parameter values. The representable range is from -16.0 to 15.9995 (15 + 2047/2048). The resultant parameters are sent from the switcher to the cameras over SDI.

#### Syntax

**HRESULT** OffsetFloats(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t count, const double\* values)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	in	The number of elements.
values	in	The parameter values.

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.
E_POINTER	The values argument is NULL.
E_OUTOFMEMORY	There was insufficient memory to complete the operation.

### 8.2.3.23 IBMDSwitcherCameraControl::SetFloats method

The **SetFloats** method sends fixed floating point values from the switcher to the cameras over SDI. The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

#### Syntax

```
HRESULT          SetFloats(uint32_t destinationDevice, uint32_t category, uint32_t parameter,
                           uint32_t count, const double* values)
```

#### Parameters

Name	Direction	Description
<code>destinationDevice</code>	in	The destination device address.
<code>category</code>	in	The category.
<code>parameter</code>	in	The parameter.
<code>count</code>	in	The element count.
<code>values</code>	in	The parameter values.

#### Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The values argument is NULL.
<code>E_OUTOFMEMORY</code>	There was insufficient memory to complete the operation.

### 8.2.3.24 IBMDSwitcherCameraControl::GetFloats method

The **GetFloats** method returns the last fixed point values sent to the cameras over SDI.  
The representable range is from -16.0 to 15.9995 (15 + 2047/2048).

#### Syntax

**HRESULT**            GetFloats(uint32\_t destinationDevice, uint32\_t category, uint32\_t parameter, uint32\_t\* count, double\* values)

#### Parameters

Name	Direction	Description
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	out	The element count.
values	out	The fixed point parameter values.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	Floats do not exist for the given arguments.
E_POINTER	The count or values argument is NULL.

### 8.2.3.25 IBMDSwitcherCameraControl::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherCameraControlCallback** object. The caller should pass an object implementing the **IBMDSwitcherCameraControlCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

**HRESULT**           AddCallback(IBMDSwitcherCameraControlCallback\* callback)

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherCameraControlCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is NULL.

### 8.2.3.26 IBMDSwitcherCameraControl::RemoveCallback method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

**HRESULT** RemoveCallback(IBMDSwitcherCameraControlCallback\* callback)

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherCameraControlCallback</b> object interface.

#### Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The callback parameter is invalid.

## 9 Macros

The Switcher Macros API provides the functionality to record, playback, and transfer macros.

### 9.1 General Information

#### 9.1.1 Macro Indexes and Identification

Each switcher is capable of storing a maximum number of macros. Each macro is uniquely identified by an index ranging from 0 to  $n - 1$ , where  $n$  is the maximum number of macros available on the switcher. A macro is stored using the index specified on record or upload. If there is already a macro with the same index then it will be replaced, so it is best to check for an existing macro at the specified index before recording or uploading. If a macro exists at the specified index then the macro is considered valid, otherwise the index contains no macro.

#### 9.1.2 Recording a Macro

Here are the basic steps for recording a macro on a switcher:

- Ensure you are connected to a switcher and have an **IBMDSwitcher** object.  
Please refer to the Basic Switcher Control section for how to do this.
- Get the **IBMDSwitcherMacroControl** interface from the **IBMDSwitcher** object.
- Create an **IBMDSwitcherMacroControlCallback** object and add it to the macro pool with the **IBMDSwitcherMacroControl::AddCallback** method.
- Call **IBMDSwitcherMacroControl::Record** to begin recording a new macro.
- Perform the switcher operations that you wish to record to the macro.
- Call **IBMDSwitcherMacroControl::StopRecording** to end the macro recording.



### 9.1.3 Downloading a Macro

Here are the basic steps for downloading a macro from a switcher:

- Ensure you are connected to a switcher and have an **IBMDSwitcher** object. Please refer to the Basic Switcher Control section for how to do this.
- Get the **IBMDSwitcherMacroPool** interface from the **IBMDSwitcher** object.
- Create an **IBMDSwitcherMacroPoolCallback** object and add it to the macro pool with the **IBMDSwitcherMacroPool::AddCallback** method.
- Call **IBMDSwitcherMacroPool::Download** to begin the transfer of a macro from the switcher. This method will return an **IBMDSwitcherTransferMacro** object which can be used to track the progress of the download. You will also be notified of the download outcome via the **IBMDSwitcherMacroPoolCallback** interface.
- Using the **IBMDSwitcherTransferMacro** object obtained from either **IBMDSwitcherMacroPool::Download** or **IBMDSwitcherMacroPoolCallback::Notify** call **IBMDSwitcherTransferMacro::GetMacro** to get the macro object.

### 9.1.4 Uploading a Macro

The steps for uploading a macro to a switcher are very similar to downloading:

- Ensure you are connected to a switcher and have an **IBMDSwitcher** object. Please refer to the Basic Switcher Control section for how to do this.
- Get the **IBMDSwitcherMacroPool** interface from the **IBMDSwitcher** object.
- Create an **IBMDSwitcherMacroPoolCallback** object and add it to the macro pool with the **IBMDSwitcherMacroPool::AddCallback** method.
- Use **IBMDSwitcherMediaPool::CreateMacro** to create a macro object. Populate this with your macro binary data by filling in the macro object's data buffer, which is available via the **IBMDSwitcherMacro::GetBytes** method. Alternatively, you can use an **IBMDSwitcherMacro** object obtained from a previous macro download.
- Call **IBMDSwitcherMacroPool::Upload** to begin the transfer of the macro to the switcher. This method will return an **IBMDSwitcherTransferMacro** object which can be used to track the progress of the upload. You will also be notified of the upload outcome via the **IBMDSwitcherMacroPoolCallback** interface.

### 9.1.5 Unsupported Operations

As the capabilities of Blackmagic Design Switcher products evolve, new functionality will be added. A macro that is created on a newer version switcher and transferred to an older version switcher may contain operations that the older switcher does not support. A macro containing unsupported operations is flagged as such and can still be played, but the unsupported operations will be ignored.

## 9.2 Macro Data Types

### 9.2.1 Macro Pool Event Type

**BMDSwitcherMacroPoolEventType** enumerates the possible event types for the **IBMDSwitcherMacroPoolCallback** object interface.

<b>bmdSwitcherMacroPoolEventTypeValidChanged</b>	A macro has been created (becomes valid), or deleted (becomes invalid).
<b>bmdSwitcherMacroPoolEventTypeHasUnsupportedOpsChanged</b>	A macro's unsupported operations flag has changed.
<b>bmdSwitcherMacroPoolEventTypeNameChanged</b>	A macro's name has changed.
<b>bmdSwitcherMacroPoolEventTypeDescriptionChanged</b>	A macro's description has changed.
<b>bmdSwitcherMacroPoolEventTypeTransferCompleted</b>	A macro transfer has completed.
<b>bmdSwitcherMacroPoolEventTypeTransferCancelled</b>	A macro transfer has cancelled.
<b>bmdSwitcherMacroPoolEventTypeTransferFailed</b>	A macro transfer has failed.

### 9.2.2 Macro Control Event Type

**BMDSwitcherMacroControlEventType** enumerates the possible event types for the **IBMDSwitcherMacroControlCallback** object interface.

<b>bmdSwitcherMacroControlEventTypeRunStatusChanged</b>	The switcher's macro playback state has changed
<b>bmdSwitcherMacroControlEventTypeRecordStatusChanged</b>	The switcher's macro record state has changed.

### 9.2.3 Macro Run Status

**BMDSwitcherMacroRunStatus** enumerates the possible macro playback states.

<b>bmdSwitcherMacroRunStatusIdle</b>	No macro is playing.
<b>bmdSwitcherMacroRunStatusRunning</b>	A macro is playing.
<b>bmdSwitcherMacroRunStatusWaitingForUser</b>	A macro is waiting for the user to continue playing.

### 9.2.4 Macro Record Status

**BMDSwitcherMacroRecordStatus** enumerates the possible macro record states.

<b>bmdSwitcherMacroRecordStatusIdle</b>	No macro is being recorded.
<b>bmdSwitcherMacroRecordStatusRecording</b>	A macro is being recorded.

9.3

Interface Reference

9.3.1

IBMDSwitcherMacroPool Interface

The **IBMDSwitcherMacroPool** object interface provides functionality for the transfer and deletion of macros and for accessing and modifying macro properties.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An <b>IBMDSwitcherMacroPool</b> object interface can be obtained with the <b>IBMDSwitcher:QueryInterface</b> method.

Public Member Functions	
Method	Description
GetMaxCount	Gets the number of macros that can be stored.
Delete	Deletes a macro.
IsValid	Checks if a macro exists.
HasUnsupportedOps	Checks if a macro has unsupported operations.
GetName	Gets a macro's name.
SetName	Sets a macro's name.
GetDescription	Gets a macro's description.
SetDescription	Sets a macro's description.
CreateMacro	Creates an <b>IBMDSwitcherMacro</b> object.
Upload	Uploads a macro.
Download	Downloads a macro.
AddCallback	Adds a macro pool callback.
RemoveCallback	Removes a macro pool callback.

### 9.3.1.1 **IBMDSwitcherMacroPool::GetMaxCount method**

The **GetMaxCount** method returns the number of macros that can be stored on the switcher.

#### Syntax

```
HRESULT GetMaxCount (uint32_t* maxCount);
```

#### Parameters

Name	Direction	Description
maxCount	out	The maximum number of macros for the switcher.

#### Return Values

Value	Description
E_POINTER	The maxCount parameter is invalid.
S_OK	Success.

### 9.3.1.2 **IBMDSwitcherMacroPool::Delete method**

The **Delete** method will delete (set invalid) an existing macro. If the macro is already invalid then this method has no effect.

#### Syntax

```
HRESULT Delete (uint32_t index);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.

#### Return Values

Value	Description
E_INVALIDARG	The index parameter is out of range.
S_OK	Success.

### 9.3.1.3 IBMDSwitcherMacroPool::IsValid method

The **IsValid** method checks if a macro with the specified index exists.

#### Syntax

```
HRESULT IsValid (uint32_t index, boolean* valid);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
valid	out	Boolean value which is true if the macro is valid.

#### Return Values

Value	Description
E_POINTER	The valid parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
S_OK	Success.

#### 9.3.1.4 **IBMDSwitcherMacroPool::HasUnsupportedOps** method

The **HasUnsupportedOps** method indicates whether a macro contains unsupported operations. A macro with unsupported operations can still be played but the unsupported operations will be ignored.

##### **Syntax**

```
HRESULT HasUnsupportedOps (uint32_t index, boolean* hasUnsupportedOps);
```

##### **Parameters**

Name	Direction	Description
index	in	Macro index.
hasUnsupportedOps	out	Boolean value which is true if the macro contains unsupported operations.

##### **Return Values**

Value	Description
E_POINTER	The hasUnsupportedOps parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
S_OK	Success.

### 9.3.1.5 IBMDSwitcherMacroPool::GetName method

The **GetName** method gets the name of a macro.

#### Syntax

```
HRESULT      GetName (uint32_t index, string* name);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
name	out	Macro name.

#### Return Values

Value	Description
E_POINTER	The name parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to get the name.
S_OK	Success.



### 9.3.1.6 IBMDSwitcherMacroPool::SetName method

The **SetName** method sets the name of a macro.

#### Syntax

```
HRESULT      SetName (uint32_t index, string name);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
name	in	Macro name.

#### Return Values

Value	Description
E_POINTER	The name parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to set the name.
S_OK	Success.

### 9.3.1.7 IBMDSwitcherMacroPool::GetDescription method

The **GetDescription** method gets the description of a macro.

#### Syntax

```
HRESULT GetDescription (uint32_t index, string* description);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
description	out	Macro description.

#### Return Values

Value	Description
E_POINTER	The description parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to get the description.
S_OK	Success.

### 9.3.1.8 **IBMDSwitcherMacroPool::SetDescription** method

The **SetDescription** method sets the description of a macro.

#### Syntax

```
HRESULT SetDescription (uint32_t index, string description);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
description	out	Macro description.

#### Return Values

Value	Description
E_POINTER	The description parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to set the description.
S_OK	Success.

### 9.3.1.9 **IBMDSwitcherMacroPool::CreateMacro** method

The **CreateMacro** method creates an **IBMDSwitcherMacro** object.  
**IBMDSwitcherMacro** objects are only used for transfers.

#### Syntax

```
HRESULT CreateMacro (uint32_t sizeBytes, IBMDSwitcherMacro** macro);
```

#### Parameters

Name	Direction	Description
sizeBytes	in	The size of the macro, in bytes.
macro	out	The <b>IBMDSwitcherMacro</b> object.

#### Return Values

Value	Description
E_OUTOFMEMORY	Insufficient memory to create a macro.
S_OK	Success.

### 9.3.1.10 **IBMDSwitcherMacroPool::Upload method**

The **Upload** method transfers a macro to the switcher. No more than one transfer can occur at a time.

#### Syntax

```
HRESULT Upload (uint32_t index, string name, string description, IBMDSwitcherMacro* macro,
                IBMDSwitcherTransferMacro** macroTransfer);
```

#### Parameters

Name	Direction	Description
index	in	Destination macro index.
name	in	Destination macro name.
description	in	Destination macro description.
macro	in	<b>IBMDSwitcherMacro</b> object containing the macro binary data for the transfer.
macroTransfer	out	<b>IBMDSwitcherMacroTransfer</b> object for monitoring the progress of the transfer.

#### Return Values

Value	Description
E_POINTER	The name, description, or macro parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to perform a transfer.
S_OK	Success.

### 9.3.1.11 IBMDSwitcherMacroPool::Download method

The **Download** method transfers a macro from the switcher. No more than one transfer can occur at a time.

#### Syntax

```
HRESULT Download (uint32_t index, IBMDSwitcherTransferMacro** macroTransfer);
```

#### Parameters

Name	Direction	Description
index	in	Destination macro index.
macroTransfer	out	<b>IBMDSwitcherMacroTransfer</b> object for monitoring the progress of the transfer and retrieving the macro binary data.

#### Return Values

Value	Description
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to perform a transfer.
S_OK	Success.

### 9.3.1.12 **IBMDSwitcherMacroPool::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherMacroPool** object. Pass an object implementing the **IBMDSwitcherMacroPoolCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

```
HRESULT AddCallback (IBMDSwitcherMacroPoolCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMacroPoolCallback</b> object interface.

#### Return Values

Value	Description
E_INVALIDARG	The callback parameter is invalid.
S_OK	Success.

### 9.3.1.13 **IBMDSwitcherMacroPool::RemoveCallback** method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherMacroPoolCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMacroPoolCallback</b> object interface.

#### Return Values

Value	Description
E_INVALIDARG	The callback parameter is invalid.
S_OK	Success.



### 9.3.2 **IBMDSwitcherTransferMacro** Interface

The **IBMDSwitcherTransferMacro** object interface provides methods to cancel a macro transfer, monitor transfer progress, and retrieve transferred macro binary data.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMacroPool	IID_IBMDSwitcherMacroPool	<b>IBMDSwitcherMacroPool::Upload</b> and <b>IBMDSwitcherMacroPool::Download</b> return an <b>IBMDSwitcherTransferMacro</b> object.
IBMDSwitcherMacroPoolCallback	IID_IBMDSwitcherMacroPool	<b>IBMDSwitcherMacroPoolCallback::Notify</b> passes in an <b>IBMDSwitcherTransferMacro</b> object.

#### Public Member Functions

Method	Description
Cancel	Cancels the pending transfer.
GetProgress	Gets the pending transfer's progress.
GetMacro	Gets an <b>IBMDSwitcherMacro</b> object from a completed transfer.

### 9.3.2.1 **IBMDSwitcherTransferMacro::Cancel** method

The **Cancel** method cancels the pending transfer. If there is no pending macro transfer then this method has no effect.

#### Syntax

```
HRESULT      Cancel (void);
```

#### Parameters

none

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 9.3.2.2 **IBMDSwitcherTransferMacro::GetProgress** method

The **GetProgress** method gets the progress of the pending transfer.

#### Syntax

```
HRESULT      GetProgress (double* progress);
```

#### Parameters

Name	Direction	Description
progress	out	Transfer progress. Range is between 0.0 to 1.0.

#### Return Values

Value	Description
E_POINTER	The progress parameter is invalid.
S_OK	Success.

### 9.3.2.3 IBMDSwitcherTransferMacro::GetMacro method

The **GetMacro** method gets the transferred **IBMDSwitcherMacro** object.

#### Syntax

```
HRESULT GetMacro (IBMDSwitcherMacro** macro);
```

#### Parameters

Name	Direction	Description
macro	out	Pointer to an <b>IBMDSwitcherMacro</b> object.

#### Return Values

Value	Description
E_POINTER	The macro parameter is invalid.
E_UNEXPECTED	No transfer has been initiated.
S_OK	Success.

### 9.3.3 **IBMDSwitcherMacro Interface**

The **IBMDSwitcherMacro** object interface provides access to macro binary data used for transferring macros.

This interface does not provide access to macro properties or control to record or playback a macro. To access properties use the **IBMDSwitcherMacroPool** interface. To record or playback a macro use the **IBMDSwitcherMacroControl** interface.

#### **Related Interfaces**

Interface	Interface ID	Description
IBMDSwitcherMacroPool	IID_IBMDSwitcherMacroPool	<b>IBMDSwitcherMacroPool::CreateMacro</b> returns an <b>IBMDSwitcherMacro</b> object.
IBMDSwitcherTransferMacro	IID_IBMDSwitcherMacroTransfer	<b>IBMDSwitcherMacroTransfer::GetMacro</b> returns an <b>IBMDSwitcherMacro</b> object.

#### **Public Member Functions**

Method	Description
GetSize	Gets the size (in bytes) of the macro binary data.
GetBytes	Gets a pointer to the macro binary data buffer.

### 9.3.3.1 IBMDSwitcherMacro::GetSize method

The **GetSize** method returns the size (in bytes) of the macro binary data.

#### Syntax

```
uint32_t    GetSize (void);
```

#### Parameters

none

#### Return Values

Value	Description
Size	Size (in bytes) of the macro binary data.

### 9.3.3.2 IBMDSwitcherMacro::GetBytes method

The **GetBytes** method returns a pointer to the macro binary data buffer.

#### Syntax

```
HRESULT    GetBytes (void** buffer);
```

#### Parameters

Name	Direction	Description
buffer	out	Pointer to the macro binary data.

#### Return Values

Value	Description
E_POINTER	The buffer parameter is invalid.
S_OK	Success.

9.3.4

IBMDSwitcherMacroPoolCallback Interface

The **IBMDSwitcherMacroPoolCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherMacroPool** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMacroPool	IID_IBMDSwitcherMacroPool	An <b>IBMDSwitcherMacroPoolCallback</b> object interface is installed with <b>IBMDSwitcherMacroPool::AddCallback</b> and removed with <b>IBMDSwitcherMacroPool::RemoveCallback</b>

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

#### 9.3.4.1 IBMDSwitcherMacroPoolCallback::Notify method

The **Notify** method is called when an **IBMDSwitcherMacroPool** event occurs, such as macro property changes.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

##### Syntax

```
HRESULT Notify (BMDSwitcherMacroPoolEventType eventType, uint32_t index,
                IBMDSwitcherTransferMacro* macroTransfer);
```

##### Parameters

Name	Direction	Description
eventType	in	<b>BMDSwitcherMacroPoolEventType</b> that describes the type of event that has occurred.
index	in	Index of the macro that has changed.
macroTransfer	in	If the event type is one of <b>bmdSwitcherMacroPoolEventTypeTransferCompleted</b> , <b>bmdSwitcherMacroPoolEventTypeTransferCancelled</b> , or <b>bmdSwitcherMacroPoolEventTypeTransferFailed</b> then this parameter is a pointer to the affected <b>IBMDSwitcherTransferMacro</b> interface object.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 9.3.5 IBMDSwitcherMacroControl Interface

The **IBMDSwitcherMacroControl** object interface provides macro recording state, playback state, and control.

#### Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An <b>IBMDSwitcherMacroControl</b> object interface can be obtained with the <b>IBMDSwitcher::QueryInterface</b> .

#### Public Member Functions

Method	Description
Run	Begins playback of a macro.
GetLoop	Gets the playback loop setting.
SetLoop	Sets the playback loop setting.
ResumeRunning	Resumes playback of a waiting macro.
StopRunning	Stops a playing or waiting macro.
Record	Begins recording of a new macro.
RecordUserWait	Inserts a user wait into the currently recording macro.
RecordPause	Inserts a time delay into the currently recording macro.
StopRecording	Ends recording.
GetRunStatus	Gets the current playback state.
GetRecordStatus	Gets the current record state.
AddCallback	Adds a macro control callback.
RemoveCallback	Removes a macro control callback.



### 9.3.5.1 IBMDSwitcherMacroControl::Run method

The **Run** method begins playback of a macro.

#### Syntax

```
HRESULT          Run (uint32_t index);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.

#### Return Values

Value	Description
E_INVALIDARG	The index parameter is out of range or invalid.
E_FAIL	Failure.
S_OK	Success.

### 9.3.5.2 **IBMDSwitcherMacroControl::GetLoop** method

The **GetLoop** method gets the current loop setting. When true, a running macro will loop back to the start when the last operation completes.

#### **Syntax**

**HRESULT**            GetLoop (boolean\* loop);

#### **Parameters**

Name	Direction	Description
loop	out	Boolean value which is true if playback will loop.

#### **Return Values**

Value	Description
E_POINTER	The loop parameter is invalid.
E_FAIL	Failure.
S_OK	Success.

### 9.3.5.3 **IBMDSwitcherMacroControl::SetLoop** method

The **SetLoop** method sets the current loop setting. When true, a running macro will loop back to the start when the last operation completes.

#### **Syntax**

**HRESULT**            **SetLoop** (boolean loop);

#### **Parameters**

Name	Direction	Description
loop	in	Desired setting for loop, which is true if playback will loop.

#### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 9.3.5.4 **IBMDSwitcherMacroControl::ResumeRunning** method

The **ResumeRunning** method continues playback of a macro that is waiting for the user. If there is no macro currently waiting then this method has no effect.

##### **Syntax**

**HRESULT**          ResumeRunning (void);

##### **Parameters**

none

##### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 9.3.5.5 **IBMDSwitcherMacroControl::StopRunning** method

The **StopRunning** method stops the currently playing macro. If there is no macro currently playing then this method has no effect.

##### **Syntax**

**HRESULT**          StopRunning (void);

##### **Parameters**

none

##### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 9.3.5.6 IBMDSwitcherMacroControl::Record method

The **Record** method begins recording of a new macro.

#### Syntax

```
HRESULT Record (uint32_t index, string name, string description);
```

#### Parameters

Name	Direction	Description
index	in	Macro index.
name	in	Name of the macro.
description	in	Description of the macro.

#### Return Values

Value	Description
E_POINTER	The name or description parameter is invalid.
E_INVALIDARG	The index parameter is out of range.
E_OUTOFMEMORY	Insufficient memory to record a new macro.
E_FAIL	Failure.
S_OK	Success.

#### 9.3.5.7 IBMDSwitcherMacroControl::RecordUserWait method

The **RecordUserWait** method inserts a user wait into the currently recording macro. If there is no macro currently recording then this method has no effect.

##### Syntax

```
HRESULT      RecordUserWait (void);
```

##### Parameters

None

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

#### 9.3.5.8 IBMDSwitcherMacroControl::RecordPause method

The **RecordPause** method inserts a timed pause into the currently recording macro. If there is no macro currently recording then this method has no effect.

##### Syntax

```
HRESULT      RecordPause (uint32_t frames);
```

##### Parameters

Name	Direction	Description
frames	in	Number of frames to pause for.

##### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 9.3.5.9 IBMDSwitcherMacroControl::StopRecording method

The **StopRecording** method stops the currently recording macro. If there is no macro currently recording then this method has no effect.

#### Syntax

```
HRESULT      StopRecording (void);
```

#### Parameters

none

#### Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

### 9.3.5.10 IBMDSwitcherMacroControl::GetRunStatus method

The **GetRunStatus** method gets the current playback state of the switcher.

#### Syntax

```
HRESULT GetRunStatus (BMDSwitcherMacroRunStatus* status, boolean* loop, uint32_t* index);
```

#### Parameters

Name	Direction	Description
status	out	<b>BMDSwitcherMacroRunStatus</b> value indicating the macro playback state.
loop	out	Boolean value which is true if playback is set to loop.
index	out	Index of the macro that is playing/waiting.

#### Return Values

Value	Description
E_POINTER	The status, loop, or index parameter is invalid.
S_OK	Success.



### 9.3.5.11 **IBMDSwitcherMacroControl::GetRecordStatus** method

The **GetRecordStatus** method gets the current record state of the switcher.

#### Syntax

```
HRESULT GetRecordStatus (BMDSwitcherMacroRecordStatus* status, uint32_t* index);
```

#### Parameters

Name	Direction	Description
status	out	<b>BMDSwitcherMacroRecordStatus</b> value indicating the macro recording state.
index	out	Index of the macro that is recording.

#### Return Values

Value	Description
E_POINTER	The status, or index parameter is invalid.
S_OK	Success.

### 9.3.5.12 **IBMDSwitcherMacroControl::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherMacroControl** object. Pass an object implementing the **IBMDSwitcherMacroControlCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

#### Syntax

**HRESULT**           AddCallback (IBMDSwitcherMacroControlCallback\* callback);

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMacroControlCallback</b> object interface.

#### Return Values

Value	Description
E_INVALIDARG	The callback parameter is invalid.
S_OK	Success.

### 9.3.5.13 **IBMDSwitcherMacroControl::RemoveCallback** method

The **RemoveCallback** method removes a previously installed callback.

#### Syntax

```
HRESULT RemoveCallback (IBMDSwitcherMacroControlCallback* callback);
```

#### Parameters

Name	Direction	Description
callback	in	Callback object implementing the <b>IBMDSwitcherMacroControlCallback</b> object interface.

#### Return Values

Value	Description
E_INVALIDARG	The callback parameter is invalid.
S_OK	Success.

9.3.6

IBMDSwitcherMacroControlCallback Interface

The **IBMDSwitcherMacroControlCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherMacroControl** object. Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMacroControl	IID_IBMDSwitcherMacroControl	An <b>IBMDSwitcherMacroControlCallback</b> object interface is installed with <b>IBMDSwitcherMacroControl::AddCallback</b> and removed with <b>IBMDSwitcherMacroControl::RemoveCallback</b> .

Public Member Functions	
Method	Description
Notify	Called when an event occurs.

### 9.3.6.1 **IBMDSwitcherMacroControlCallback::Notify method**

The **Notify** method is called when an **IBMDSwitcherMacroControl** event occurs, such as a macro playback and recording states.

This method is called from a separate thread created by the switcher SDK so care should be exercised when interacting with other threads. Callbacks should be processed as quickly as possible to avoid delaying other callbacks or affecting the connection to the switcher.

The return value (required by COM) is ignored by the caller.

#### **Syntax**

**HRESULT** Notify (BMDSwitcherMacroControlEventType eventType);

#### **Parameters**

Name	Direction	Description
eventType	in	<b>BMDSwitcherMacroControlEventType</b> that describes the type of event that has occurred.

#### **Return Values**

Value	Description
E_FAIL	Failure.
S_OK	Success.

