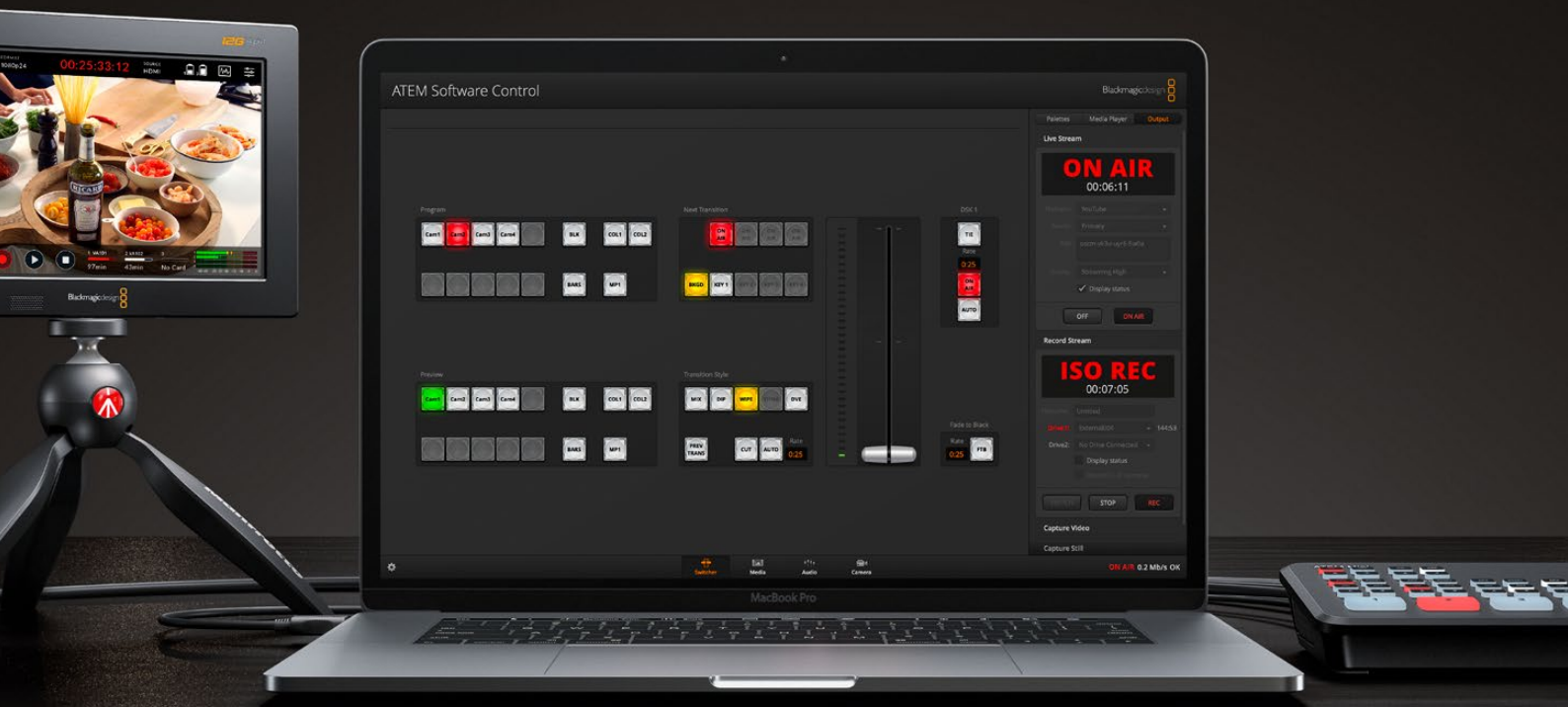


ATEM Switchers SDK



Contents

Introduction	42
Welcome	42
Overview	42
Section 1 — API Design	42
1.1 Overview	42
1.2 Object Model	42
1.3 Object Interfaces	43
1.4 Reference Counting	43
1.5 Interface Stability	43
1.5.1 New Interfaces	43
1.5.2 Updated Interfaces	43
1.5.3 Deprecated Interfaces	44
1.5.4 Removed Interfaces	44
1.6 Interface Reference	44
1.6.1 IUnknown Interface	44
1.6.1.1 IUnknown::QueryInterface method	44
1.6.1.2 IUnknown::AddRef method	45
1.6.1.3 IUnknown::Release method	45
1.7 Using the Switcher API in a project	45
1.7.1 Basic Types	45
1.7.2 Accessing Switcher devices	47
1.7.2.1 Windows	47
1.7.2.2 macOS OS	47
Section 2 — Basic Switcher Control	48
2.1 General Information	48
2.1.1 Switcher Configuration and Transitions	48
2.1.2 Switcher Interface Diagram	48
2.2 Switcher Data Types	49
2.2.1 Basic Switcher Data Types	49
2.2.2 Switcher Event Type	49
2.2.3 Switcher Power Status	49
2.2.4 Switcher Video Mode	50
2.2.5 Switcher Down Conversion Methods	51
2.2.6 Switcher Input Event Type	51
2.2.7 Switcher External Port Types	52
2.2.8 Switcher Port Types	52
2.2.9 Switcher Input Availability	53
2.2.10 Switcher Mix Effect Block Events	53
2.2.11 Switcher Connection Errors	54

2.2.12	Switcher MultiView Layouts	54
2.2.13	Switcher MultiView Safe Area Type	55
2.2.14	Switcher Serial Port Functions	55
2.2.15	Switcher Serial Port Baud Rate	55
2.2.16	Switcher 3G-SDI Output Levels	55
2.2.17	Switcher Colorimetry Mode	55
2.2.18	Switcher Mix Minus Output Audio Modes	56
2.2.19	Switcher Color Events	56
2.2.20	Switcher Aux Events	56
2.2.21	Switcher MultiView Events	56
2.2.22	Switcher Serial Port Events	57
2.2.23	Switcher Mix Minus Output Events	57
2.2.24	Save And Recall Type	57
2.2.25	Switcher TimeCode Mode Type	57
2.3	Interface Reference	58
2.3.1	IBMDSwitcherDiscovery Interface	58
2.3.1.1	IBMDSwitcherDiscovery::ConnectTo method	58
2.3.2	IBMDSwitcher Interface	59
2.3.2.1	IBMDSwitcher::GetProductName method	60
2.3.2.2	IBMDSwitcher::GetVideoMode method	61
2.3.2.3	IBMDSwitcher::SetVideoMode method	61
2.3.2.4	IBMDSwitcher::DoesSupportVideoMode method	62
2.3.2.5	IBMDSwitcher::DoesVideoModeChangeRequireReconfiguration method	62
2.3.2.6	IBMDSwitcher::GetMethodForDownConvertedSD method	63
2.3.2.7	IBMDSwitcher::SetMethodForDownConvertedSD method	63
2.3.2.8	IBMDSwitcher::GetDownConvertedHDVideoMode method	64
2.3.2.9	IBMDSwitcher::SetDownConvertedHDVideoMode method	64
2.3.2.10	IBMDSwitcher::DoesSupportDownConvertedHDVideoMode method	65
2.3.2.11	IBMDSwitcher::GetMultiViewVideoMode method	65
2.3.2.12	IBMDSwitcher::SetMultiViewVideoMode method	66
2.3.2.13	IBMDSwitcher::Get3GSDIOutputLevel method	66
2.3.2.14	IBMDSwitcher::Set3GSDIOutputLevel method	67
2.3.2.15	IBMDSwitcher::DoesSupportColorimetrySetting method	67
2.3.2.16	IBMDSwitcher::GetColorimetryMode method	68
2.3.2.17	IBMDSwitcher::SetColorimetryMode method	68
2.3.2.18	IBMDSwitcher::DoesSupportMultiViewVideoMode method	69
2.3.2.19	IBMDSwitcher::GetPowerStatus method	69
2.3.2.20	IBMDSwitcher::GetTimeCode method	70
2.3.2.21	IBMDSwitcher::SetTimeCode method	70
2.3.2.22	IBMDSwitcher::RequestTimeCode method	71
2.3.2.23	IBMDSwitcher::GetTimeCodeLocked method	71
2.3.2.24	IBMDSwitcher::GetTimeCodeMode method	71
2.3.2.25	IBMDSwitcher::SetTimeCodeMode method	72
2.3.2.26	IBMDSwitcher::GetAreOutputsConfigurable method	72
2.3.2.27	IBMDSwitcher::GetSuperSourceCascade method	73

2.3.2.28	IBMDSwitcher::SetSuperSourceCascade method	73
2.3.2.29	IBMDSwitcher::SuspendStreaming method	74
2.3.2.30	IBMDSwitcher::AllowStreamingToResume method	74
2.3.2.31	IBMDSwitcher::DoesSupportFadeToBlackEnabledSetting method	74
2.3.2.32	IBMDSwitcher::GetFadeToBlackEnabled method	75
2.3.2.33	IBMDSwitcher::SetFadeToBlackEnabled method	75
2.3.2.34	IBMDSwitcher::DoesSupportAutoVideoMode method	76
2.3.2.35	IBMDSwitcher::GetAutoVideoMode method	76
2.3.2.36	IBMDSwitcher::GetAutoVideoModeDetected method	77
2.3.2.37	IBMDSwitcher::SetAutoVideoMode method	77
2.3.2.38	IBMDSwitcher::DoesSupportDskTallyOverride method	78
2.3.2.39	IBMDSwitcher::GetDskTallyOverrideEnabled method	78
2.3.2.40	IBMDSwitcher::SetDskTallyOverrideEnabled method	79
2.3.2.41	IBMDSwitcher::CreateIterator method	79
2.3.2.42	IBMDSwitcher::AddCallback method	80
2.3.2.43	IBMDSwitcher::RemoveCallback method	80
2.3.3	IBMDSwitcherCallback Interface	81
2.3.3.1	IBMDSwitcherCallback::Notify method	81
2.3.4	IBMDSwitcherInputIterator Interface	82
2.3.4.1	IBMDSwitcherInputIterator::Next method	82
2.3.4.2	IBMDSwitcherInputIterator::GetById method	83
2.3.5	IBMDSwitcherInput Interface	83
2.3.5.1	IBMDSwitcherInput::AddCallback method	84
2.3.5.2	IBMDSwitcherInput::RemoveCallback method	85
2.3.5.3	IBMDSwitcherInput::GetInputId method	85
2.3.5.4	IBMDSwitcherInput::GetPortType method	86
2.3.5.5	IBMDSwitcherInput::GetInputAvailability method	86
2.3.5.6	IBMDSwitcherInput::SetShortName method	87
2.3.5.7	IBMDSwitcherInput::GetShortName method	87
2.3.5.8	IBMDSwitcherInput::SetLongName method	88
2.3.5.9	IBMDSwitcherInput::GetLongName method	88
2.3.5.10	IBMDSwitcherInput::AreNamesDefault method	89
2.3.5.11	IBMDSwitcherInput::ResetNames method	89
2.3.5.12	IBMDSwitcherInput::IsProgramTallied method	89
2.3.5.13	IBMDSwitcherInput::IsPreviewTallied method	90
2.3.5.14	IBMDSwitcherInput::GetAvailableExternalPortTypes method	90
2.3.5.15	IBMDSwitcherInput::SetCurrentExternalPortType method	91
2.3.5.16	IBMDSwitcherInput::DoesSupportCameraModel method	91
2.3.5.17	IBMDSwitcherInput::SetCameraModel method	92
2.3.5.18	IBMDSwitcherInput::GetCameraModel method	92
2.3.5.19	IBMDSwitcherInput::SetViscaDeviceId method	93
2.3.5.20	IBMDSwitcherInput::GetViscaDeviceId method	93
2.3.6	IBMDSwitcherInputCallback Interface	94
2.3.6.1	IBMDSwitcherInputCallback::Notify method	94
2.3.7	IBMDSwitcherMixEffectBlockIterator Interface	95

2.3.7.1	IBMDSwitcherMixEffectBlockIterator::Next method	95
2.3.8	IBMDSwitcherMixEffectBlock Interface	96
2.3.8.1	IBMDSwitcherMixEffectBlock::GetProgramInput method	97
2.3.8.2	IBMDSwitcherMixEffectBlock::SetProgramInput method	97
2.3.8.3	IBMDSwitcherMixEffectBlock::GetPreviewInput method	98
2.3.8.4	IBMDSwitcherMixEffectBlock::SetPreviewInput method	98
2.3.8.5	IBMDSwitcherMixEffectBlock::GetPreviewLive method	99
2.3.8.6	IBMDSwitcherMixEffectBlock::GetPreviewTransition method	99
2.3.8.7	IBMDSwitcherMixEffectBlock::SetPreviewTransition method	99
2.3.8.8	IBMDSwitcherMixEffectBlock::PerformAutoTransition method	100
2.3.8.9	IBMDSwitcherMixEffectBlock::PerformCut method	100
2.3.8.10	IBMDSwitcherMixEffectBlock::GetInTransition method	100
2.3.8.11	IBMDSwitcherMixEffectBlock::GetTransitionPosition method	101
2.3.8.12	IBMDSwitcherMixEffectBlock::SetTransitionPosition method	101
2.3.8.13	IBMDSwitcherMixEffectBlock::GetTransitionFramesRemaining method	101
2.3.8.14	IBMDSwitcherMixEffectBlock::PerformFadeToBlack method	102
2.3.8.15	IBMDSwitcherMixEffectBlock::GetFadeToBlackRate method	102
2.3.8.16	IBMDSwitcherMixEffectBlock::SetFadeToBlackRate method	103
2.3.8.17	IBMDSwitcherMixEffectBlock::GetFadeToBlackFramesRemaining method	103
2.3.8.18	IBMDSwitcherMixEffectBlock::GetFadeToBlackFullyBlack method	103
2.3.8.19	IBMDSwitcherMixEffectBlock::SetFadeToBlackFullyBlack method	104
2.3.8.20	IBMDSwitcherMixEffectBlock::GetInFadeToBlack method	104
2.3.8.21	IBMDSwitcherMixEffectBlock::GetFadeToBlackInTransition method	104
2.3.8.22	IBMDSwitcherMixEffectBlock::GetInputAvailabilityMask method	105
2.3.8.23	IBMDSwitcherMixEffectBlock::CreateIterator method	105
2.3.8.24	IBMDSwitcherMixEffectBlock::AddCallback method	106
2.3.8.25	IBMDSwitcherMixEffectBlock::RemoveCallback method	106
2.3.9	IBMDSwitcherMixEffectBlockCallback Interface	107
2.3.9.1	IBMDSwitcherMixEffectBlockCallback::Notify method	107
2.3.10	IBMDSwitcherInputColor Interface	108
2.3.10.1	IBMDSwitcherInputColor::GetHue method	108
2.3.10.2	IBMDSwitcherInputColor::SetHue method	109
2.3.10.3	IBMDSwitcherInputColor::GetSaturation method	109
2.3.10.4	IBMDSwitcherInputColor::SetSaturation method	109
2.3.10.5	IBMDSwitcherInputColor::GetLuma method	110
2.3.10.6	IBMDSwitcherInputColor::SetLuma method	110
2.3.10.7	IBMDSwitcherInputColor::AddCallback method	111
2.3.10.8	IBMDSwitcherInputColor::RemoveCallback method	111
2.3.11	IBMDSwitcherInputColorCallback Interface	112
2.3.11.1	IBMDSwitcherInputColorCallback::Notify method	112
2.3.12	IBMDSwitcherInputAux Interface	113
2.3.12.1	IBMDSwitcherInputAux::GetInputSource method	113
2.3.12.2	IBMDSwitcherInputAux::SetInputSource method	114
2.3.12.3	IBMDSwitcherInputAux::GetInputAvailabilityMask method	114
2.3.12.4	IBMDSwitcherInputAux::AddCallback method	115

2.3.12.5	IBMDSwitcherInputAux::RemoveCallback method	115
2.3.13	IBMDSwitcherInputAuxCallback Interface	116
2.3.13.1	IBMDSwitcherInputAuxCallback::Notify method	116
2.3.14	IBMDSwitcherMultiViewIterator Interface	117
2.3.14.1	IBMDSwitcherMultiViewIterator::Next method	117
2.3.15	IBMDSwitcherMultiView Interface	118
2.3.15.1	IBMDSwitcherMultiView::CanChangeLayout method	119
2.3.15.2	IBMDSwitcherMultiView::GetLayout method	120
2.3.15.3	IBMDSwitcherMultiView::SetLayout method	120
2.3.15.4	IBMDSwitcherMultiView::SupportsQuadrantLayout method	121
2.3.15.5	IBMDSwitcherMultiView::GetWindowInput method	121
2.3.15.6	IBMDSwitcherMultiView::SetWindowInput method	122
2.3.15.7	IBMDSwitcherMultiView::GetWindowCount method	122
2.3.15.8	IBMDSwitcherMultiView::GetInputAvailabilityMask method	123
2.3.15.9	IBMDSwitcherMultiView::CanRouteInputs method	123
2.3.15.10	IBMDSwitcherMultiView::SupportsVuMeters method	124
2.3.15.11	IBMDSwitcherMultiView::CurrentInputSupportsVuMeter method	124
2.3.15.12	IBMDSwitcherMultiView::GetVuMeterEnabled method	125
2.3.15.13	IBMDSwitcherMultiView::SetVuMeterEnabled method	125
2.3.15.14	IBMDSwitcherMultiView::CanAdjustVuMeterOpacity method	126
2.3.15.15	IBMDSwitcherMultiView::GetVuMeterOpacity method	126
2.3.15.16	IBMDSwitcherMultiView::SetVuMeterOpacity method	127
2.3.15.17	IBMDSwitcherMultiView::CanToggleSafeAreaEnabled method	127
2.3.15.18	IBMDSwitcherMultiView::CurrentInputSupportsSafeArea method	128
2.3.15.19	IBMDSwitcherMultiView::GetSafeAreaEnabled method	128
2.3.15.20	IBMDSwitcherMultiView::SetSafeAreaEnabled method	129
2.3.15.21	IBMDSwitcherMultiView::GetSupportedSafeAreaTypes method	129
2.3.15.22	IBMDSwitcherMultiView::GetSafeAreaType method	130
2.3.15.23	IBMDSwitcherMultiView::SetSafeAreaType method	130
2.3.15.24	IBMDSwitcherMultiView::SupportsProgramPreviewSwap method	131
2.3.15.25	IBMDSwitcherMultiView::GetProgramPreviewSwapped method	131
2.3.15.26	IBMDSwitcherMultiView::SetProgramPreviewSwapped method	132
2.3.15.27	IBMDSwitcherMultiView::CanChangeOverlayProperties method	132
2.3.15.28	IBMDSwitcherMultiView::CurrentInputSupportsLabelOverlay method	133
2.3.15.29	IBMDSwitcherMultiView::GetLabelVisible method	133
2.3.15.30	IBMDSwitcherMultiView::SetLabelVisible method	134
2.3.15.31	IBMDSwitcherMultiView::GetBorderVisible method	134
2.3.15.32	IBMDSwitcherMultiView::SetBorderVisible method	135
2.3.15.33	IBMDSwitcherMultiView::GetBorderColor method	135
2.3.15.34	IBMDSwitcherMultiView::SetBorderColor method	136
2.3.15.35	IBMDSwitcherMultiView::AddCallback method	136
2.3.15.36	IBMDSwitcherMultiView::RemoveCallback method	137
2.3.16	IBMDSwitcherMultiViewCallback Interface	137
2.3.16.1	IBMDSwitcherMultiViewCallback::Notify method	138
2.3.17	IBMDSwitcherSerialPortIterator Interface	138

2.3.17.1	IBMDSwitcherSerialPortIterator::Next method	139
2.3.18	IBMDSwitcherSerialPort Interface	139
2.3.18.1	IBMDSwitcherSerialPort::SetFunction method	140
2.3.18.2	IBMDSwitcherSerialPort::GetFunction method	140
2.3.18.3	IBMDSwitcherSerialPort::DoesSupportFunction method	141
2.3.18.4	IBMDSwitcherSerialPort::SetBaudRate method	141
2.3.18.5	IBMDSwitcherSerialPort::GetBaudRate method	142
2.3.18.6	IBMDSwitcherSerialPort::AddCallback method	142
2.3.18.7	IBMDSwitcherSerialPort::RemoveCallback method	143
2.3.19	IBMDSwitcherSerialPortCallback Interface	143
2.3.19.1	IBMDSwitcherSerialPortCallback::Notify method	144
2.3.20	IBMDSwitcherMixMinusOutputIterator Interface	144
2.3.20.1	IBMDSwitcherMixMinusOutputIterator::Next method	145
2.3.21	IBMDSwitcherMixMinusOutput Interface	145
2.3.21.1	IBMDSwitcherMixMinusOutput::GetAvailableAudioModes method	146
2.3.21.2	IBMDSwitcherMixMinusOutput::GetAudioMode method	146
2.3.21.3	IBMDSwitcherMixMinusOutput::SetAudioMode method	147
2.3.21.4	IBMDSwitcherMixMinusOutput::HasMinusAudioInputId method	147
2.3.21.5	IBMDSwitcherMixMinusOutput::GetMinusAudioInputId method	148
2.3.21.6	IBMDSwitcherMixMinusOutput::AddCallback method	148
2.3.21.7	IBMDSwitcherMixMinusOutput::RemoveCallback method	149
2.3.22	IBMDSwitcherMixMinusOutputCallback Interface	149
2.3.22.1	IBMDSwitcherMixMinusOutputCallback::Notify method	150
2.3.23	IBMDSwitcherSaveRecall Interface	150
2.3.23.1	IBMDSwitcherSaveRecall::Save method	151
2.3.23.2	IBMDSwitcherSaveRecall::Clear method	151

Section 3 — Advanced Transitions 152

3.1	Data Types	152
3.1.1	Mix Parameters Event Type	152
3.1.2	Dip Parameters Event Type	152
3.1.3	Wipe Parameters Event Type	152
3.1.4	DVE Parameters Event Type	153
3.1.5	Stinger Parameters Event Type	153
3.1.6	Transition Parameters Event Type	154
3.1.7	Transition Style	154
3.1.8	Transition Selection	154
3.1.9	DVE Transition Style	154
3.1.10	Stinger Transition Source	155
3.2	Interface Reference	156
3.2.1	IBMDSwitcherTransitionMixParameters Interface	156
3.2.1.1	IBMDSwitcherTransitionMixParameters::GetRate method	156
3.2.1.2	IBMDSwitcherTransitionMixParameters::SetRate method	157
3.2.1.3	IBMDSwitcherTransitionMixParameters::AddCallback method	157
3.2.1.4	IBMDSwitcherTransitionMixParameters::RemoveCallback method	158

3.2.2	IBMDSwitcherTransitionMixParametersCallback Interface	158
3.2.2.1	IBMDSwitcherTransitionMixParametersCallback::Notify method	159
3.2.3	IBMDSwitcherTransitionDipParameters Interface	159
3.2.3.1	IBMDSwitcherTransitionDipParameters::GetRate method	160
3.2.3.2	IBMDSwitcherTransitionDipParameters::SetRate method	160
3.2.3.3	IBMDSwitcherTransitionDipParameters::GetInputDip method	160
3.2.3.4	IBMDSwitcherTransitionDipParameters::SetInputDip method	161
3.2.3.5	IBMDSwitcherTransitionDipParameters::AddCallback method	161
3.2.3.6	IBMDSwitcherTransitionDipParameters::RemoveCallback method	162
3.2.4	IBMDSwitcherTransitionDipParametersCallback Interface	162
3.2.4.1	IBMDSwitcherTransitionDipParametersCallback::Notify method	163
3.2.5	IBMDSwitcherTransitionWipeParametersCallback Interface	163
3.2.5.1	IBMDSwitcherTransitionWipeParametersCallback::Notify method	164
3.2.6	IBMDSwitcherTransitionWipeParameters Interface	164
3.2.6.1	IBMDSwitcherTransitionWipeParameters::GetRate method	165
3.2.6.2	IBMDSwitcherTransitionWipeParameters::SetRate method	166
3.2.6.3	IBMDSwitcherTransitionWipeParameters::GetPattern method	166
3.2.6.4	IBMDSwitcherTransitionWipeParameters::SetPattern method	167
3.2.6.5	IBMDSwitcherTransitionWipeParameters::GetBorderSize method	167
3.2.6.6	IBMDSwitcherTransitionWipeParameters::SetBorderSize method	167
3.2.6.7	IBMDSwitcherTransitionWipeParameters::GetInputBorder method	168
3.2.6.8	IBMDSwitcherTransitionWipeParameters::SetInputBorder method	168
3.2.6.9	IBMDSwitcherTransitionWipeParameters::GetSymmetry method	168
3.2.6.10	IBMDSwitcherTransitionWipeParameters::SetSymmetry method	169
3.2.6.11	IBMDSwitcherTransitionWipeParameters::GetSoftness method	169
3.2.6.12	IBMDSwitcherTransitionWipeParameters::SetSoftness method	169
3.2.6.13	IBMDSwitcherTransitionWipeParameters::GetHorizontalOffset method	170
3.2.6.14	IBMDSwitcherTransitionWipeParameters::SetHorizontalOffset method	170
3.2.6.15	IBMDSwitcherTransitionWipeParameters::GetVerticalOffset method	170
3.2.6.16	IBMDSwitcherTransitionWipeParameters::SetVerticalOffset method	171
3.2.6.17	IBMDSwitcherTransitionWipeParameters::GetReverse method	171
3.2.6.18	IBMDSwitcherTransitionWipeParameters::SetReverse method	171
3.2.6.19	IBMDSwitcherTransitionWipeParameters::GetFlipFlop method	172
3.2.6.20	IBMDSwitcherTransitionWipeParameters::SetFlipFlop method	172
3.2.6.21	IBMDSwitcherTransitionWipeParameters::AddCallback method	173
3.2.6.22	IBMDSwitcherTransitionWipeParameters::RemoveCallback method	173
3.2.7	IBMDSwitcherTransitionDVEParameters Interface	174
3.2.7.1	IBMDSwitcherTransitionDVEParameters::GetRate method	175
3.2.7.2	IBMDSwitcherTransitionDVEParameters::SetRate method	175
3.2.7.3	IBMDSwitcherTransitionDVEParameters::GetLogoRate method	176
3.2.7.4	IBMDSwitcherTransitionDVEParameters::SetLogoRate method	176
3.2.7.5	IBMDSwitcherTransitionDVEParameters::GetReverse method	176
3.2.7.6	IBMDSwitcherTransitionDVEParameters::SetReverse method	177
3.2.7.7	IBMDSwitcherTransitionDVEParameters::GetFlipFlop method	177
3.2.7.8	IBMDSwitcherTransitionDVEParameters::SetFlipFlop method	177

3.2.7.9	IBMDSwitcherTransitionDVEParameters::GetStyle method	178
3.2.7.10	IBMDSwitcherTransitionDVEParameters::SetStyle method	178
3.2.7.11	IBMDSwitcherTransitionDVEParameters::DoesSupportStyle method	179
3.2.7.12	IBMDSwitcherTransitionDVEParameters::GetNumSupportedStyles method	179
3.2.7.13	IBMDSwitcherTransitionDVEParameters::GetSupportedStyles method	180
3.2.7.14	IBMDSwitcherTransitionDVEParameters::GetInputFill method	180
3.2.7.15	IBMDSwitcherTransitionDVEParameters::SetInputFill method	181
3.2.7.16	IBMDSwitcherTransitionDVEParameters::GetInputCut method	181
3.2.7.17	IBMDSwitcherTransitionDVEParameters::SetInputCut method	182
3.2.7.18	IBMDSwitcherTransitionDVEParameters::GetFillInputAvailabilityMask method	182
3.2.7.19	IBMDSwitcherTransitionDVEParameters::GetCutInputAvailabilityMask method	183
3.2.7.20	IBMDSwitcherTransitionDVEParameters::GetEnableKey method	183
3.2.7.21	IBMDSwitcherTransitionDVEParameters::SetEnableKey method	184
3.2.7.22	IBMDSwitcherTransitionDVEParameters::GetPreMultiplied method	184
3.2.7.23	IBMDSwitcherTransitionDVEParameters::GetPreMultiplied method	184
3.2.7.24	IBMDSwitcherTransitionDVEParameters::GetClip method	185
3.2.7.25	IBMDSwitcherTransitionDVEParameters::SetClip method	185
3.2.7.26	IBMDSwitcherTransitionDVEParameters::GetGain method	185
3.2.7.27	IBMDSwitcherTransitionDVEParameters::SetGain method	186
3.2.7.28	IBMDSwitcherTransitionDVEParameters::GetInverse method	186
3.2.7.29	IBMDSwitcherTransitionDVEParameters::SetInverse method	186
3.2.7.30	IBMDSwitcherTransitionDVEParameters::AddCallback method	187
3.2.7.31	IBMDSwitcherTransitionDVEParameters::RemoveCallback method	187
3.2.8	IBMDSwitcherTransitionDVEParametersCallback Interface	188
3.2.8.1	IBMDSwitcherTransitionDVEParametersCallback::Notify method	188
3.2.9	IBMDSwitcherTransitionStingerParameters Interface	189
3.2.9.1	IBMDSwitcherTransitionStingerParameters::GetSource method	190
3.2.9.2	IBMDSwitcherTransitionStingerParameters::SetSource method	190
3.2.9.3	IBMDSwitcherTransitionStingerParameters::GetPreMultiplied method	190
3.2.9.4	IBMDSwitcherTransitionStingerParameters::SetPreMultiplied method	191
3.2.9.5	IBMDSwitcherTransitionStingerParameters::GetClip method	191
3.2.9.6	IBMDSwitcherTransitionStingerParameters::SetClip method	191
3.2.9.7	IBMDSwitcherTransitionStingerParameters::GetGain method	192
3.2.9.8	IBMDSwitcherTransitionStingerParameters::SetGain method	192
3.2.9.9	IBMDSwitcherTransitionStingerParameters::GetInverse method	192
3.2.9.10	IBMDSwitcherTransitionStingerParameters::SetInverse method	193
3.2.9.11	IBMDSwitcherTransitionStingerParameters::GetPreroll method	193
3.2.9.12	IBMDSwitcherTransitionStingerParameters::SetPreroll method	193
3.2.9.13	IBMDSwitcherTransitionStingerParameters::GetClipDuration method	194
3.2.9.14	IBMDSwitcherTransitionStingerParameters::SetClipDuration method	194
3.2.9.15	IBMDSwitcherTransitionStingerParameters::GetTriggerPoint method	194
3.2.9.16	IBMDSwitcherTransitionStingerParameters::SetTriggerPoint method	195
3.2.9.17	IBMDSwitcherTransitionStingerParameters::GetMixRate method	195
3.2.9.18	IBMDSwitcherTransitionStingerParameters::SetMixRate method	195
3.2.9.19	IBMDSwitcherTransitionStingerParameters::AddCallback method	196

3.2.9.20	IBMDSwitcherTransitionStingerParameters::RemoveCallback method	196
3.2.10	IBMDSwitcherTransitionStingerParametersCallback Interface	197
3.2.10.1	IBMDSwitcherTransitionStingerParametersCallback::Notify method	197
3.2.11	IBMDSwitcherTransitionParameters Interface	198
3.2.11.1	IBMDSwitcherTransitionParameters::GetTransitionStyle method	198
3.2.11.2	IBMDSwitcherTransitionParameters::GetNextTransitionStyle method	199
3.2.11.3	IBMDSwitcherTransitionParameters::SetNextTransitionStyle method	199
3.2.11.4	IBMDSwitcherTransitionParameters::GetTransitionSelection method	199
3.2.11.5	IBMDSwitcherTransitionParameters::SetNextTransitionSelection method	200
3.2.11.6	IBMDSwitcherTransitionParameters::GetNextTransitionSelection method	200
3.2.11.7	IBMDSwitcherTransitionParameters::AddCallback method	201
3.2.11.8	IBMDSwitcherTransitionParameters::RemoveCallback method	201
3.2.12	IBMDSwitcherTransitionParametersCallback Interface	202
3.2.12.1	IBMDSwitcherTransitionParametersCallback::Notify method	202

Section 4 — Switcher Media 203

4.1	General Information	203
4.1.1	Uploading a Still or Clip	203
4.1.2	Downloading a Still or Clip	203
4.2	Media Data Types	204
4.2.1	Switcher Pixel Format	204
4.2.2	Media Player Source Type	204
4.2.3	Media Pool Event Type	204
4.2.4	Still Capture Event Type	205
4.3	Interface Reference	205
4.3.1	IBMDSwitcherMediaPlayerCallback Interface	205
4.3.1.1	IBMDSwitcherMediaPlayerCallback::SourceChanged method	205
4.3.1.2	IBMDSwitcherMediaPlayerCallback::PlayingChanged method	206
4.3.1.3	IBMDSwitcherMediaPlayerCallback::LoopChanged method	206
4.3.1.4	IBMDSwitcherMediaPlayerCallback::AtBeginningChanged method	206
4.3.1.5	IBMDSwitcherMediaPlayerCallback::ClipFrameChanged method	207
4.3.2	IBMDSwitcherMediaPlayerIterator Interface	207
4.3.2.1	IBMDSwitcherMediaPlayerIterator::Next method	207
4.3.3	IBMDSwitcherMediaPlayer Interface	208
4.3.3.1	IBMDSwitcherMediaPlayer::GetSource method	208
4.3.3.2	IBMDSwitcherMediaPlayer::SetSource method	209
4.3.3.3	IBMDSwitcherMediaPlayer::GetPlaying method	209
4.3.3.4	IBMDSwitcherMediaPlayer::SetPlaying method	210
4.3.3.5	IBMDSwitcherMediaPlayer::GetLoop method	210
4.3.3.6	IBMDSwitcherMediaPlayer::SetLoop method	210
4.3.3.7	IBMDSwitcherMediaPlayer::GetAtBeginning method	211
4.3.3.8	IBMDSwitcherMediaPlayer::SetAtBeginning method	211
4.3.3.9	IBMDSwitcherMediaPlayer::GetClipFrame method	211
4.3.3.10	IBMDSwitcherMediaPlayer::SetClipFrame method	212
4.3.3.11	IBMDSwitcherMediaPlayer::AddCallback method	212

4.3.3.12	IBMDSwitcherMediaPlayer::RemoveCallback method	213
4.3.4	IBMDSwitcherFrame Interface	213
4.3.4.1	IBMDSwitcherFrame::GetWidth method	213
4.3.4.2	IBMDSwitcherFrame::GetHeight method	214
4.3.4.3	IBMDSwitcherFrame::GetRowBytes method	214
4.3.4.4	IBMDSwitcherFrame::GetPixelFormat method	214
4.3.4.5	IBMDSwitcherFrame::GetBytes method	215
4.3.5	IBMDSwitcherAudio Interface	215
4.3.5.1	IBMDSwitcherFrame::GetSize method	215
4.3.5.2	IBMDSwitcherAudio::GetBytes method	216
4.3.6	IBMDSwitcherLockCallback Interface	216
4.3.6.1	IBMDSwitcherLockCallback::Obtained method	216
4.3.7	IBMDSwitcherStillsCallback Interface	217
4.3.7.1	IBMDSwitcherStillsCallback::Notify method	217
4.3.8	IBMDSwitcherStills Interface	218
4.3.8.1	IBMDSwitcherStills::GetCount method	218
4.3.8.2	IBMDSwitcherStills::IsValid method	219
4.3.8.3	IBMDSwitcherStills::GetName method	219
4.3.8.4	IBMDSwitcherStills::SetName method	220
4.3.8.5	IBMDSwitcherStills::GetHash method	220
4.3.8.6	IBMDSwitcherStills::SetInvalid method	221
4.3.8.7	IBMDSwitcherStills::Lock method	221
4.3.8.8	IBMDSwitcherStills::Unlock method	222
4.3.8.9	IBMDSwitcherStills::Upload method	222
4.3.8.10	IBMDSwitcherStills::Download method	223
4.3.8.11	IBMDSwitcherStills::CancelTransfer method	223
4.3.8.12	IBMDSwitcherStills::GetProgress method	224
4.3.8.13	IBMDSwitcherStills::AddCallback method	224
4.3.8.14	IBMDSwitcherStills::RemoveCallback method	225
4.3.9	IBMDSwitcherStillCaptureCallback Interface	225
4.3.9.1	IBMDSwitcherStillCaptureCallback::Notify method	226
4.3.10	IBMDSwitcherStillCapture Interface	226
4.3.10.1	IBMDSwitcherStillCapture::IsAvailable method	227
4.3.10.2	IBMDSwitcherStillCapture::CaptureStill method	227
4.3.10.3	IBMDSwitcherStillCapture::AddCallback method	227
4.3.10.4	IBMDSwitcherStillCapture::RemoveCallback method	228
4.3.11	IBMDSwitcherClipCallback Interface	228
4.3.11.1	IBMDSwitcherClipCallback::Notify method	229
4.3.12	IBMDSwitcherClip Interface	230
4.3.12.1	IBMDSwitcherClip::GetIndex method	231
4.3.12.2	IBMDSwitcherClip::IsValid method	231
4.3.12.3	IBMDSwitcherClip::GetName method	232
4.3.12.4	IBMDSwitcherClip::SetName method	232
4.3.12.5	IBMDSwitcherClip::SetValid method	233
4.3.12.6	IBMDSwitcherClip::SetInvalid method	233

4.3.12.7	IBMDSwitcherClip::GetFrameCount method	234
4.3.12.8	IBMDSwitcherClip::GetMaxFrameCount method	234
4.3.12.9	IBMDSwitcherClip::IsFrameValid method	235
4.3.12.10	IBMDSwitcherClip::GetFrameHash method	235
4.3.12.11	IBMDSwitcherClip::IsAudioValid method	236
4.3.12.12	IBMDSwitcherClip::GetAudioName method	236
4.3.12.13	IBMDSwitcherClip::SetAudioName method	237
4.3.12.14	IBMDSwitcherClip::GetAudioHash method	237
4.3.12.15	IBMDSwitcherClip::SetAudioInvalid method	238
4.3.12.16	IBMDSwitcherClip::Lock method	238
4.3.12.17	IBMDSwitcherClip::Unlock method	239
4.3.12.18	IBMDSwitcherClip::UploadFrame method	239
4.3.12.19	IBMDSwitcherClip::DownloadFrame method	240
4.3.12.20	IBMDSwitcherClip::UploadAudio method	240
4.3.12.21	IBMDSwitcherClip::DownloadAudio method	241
4.3.12.22	IBMDSwitcherClip::CancelTransfer method	241
4.3.12.23	IBMDSwitcherClip::GetProgress method	241
4.3.12.24	IBMDSwitcherClip::AddCallback method	242
4.3.12.25	IBMDSwitcherClip::RemoveCallback method	242
4.3.13	IBMDSwitcherMediaPoolCallback Interface	243
4.3.13.1	IBMDSwitcherMediaPoolCallback::ClipFrameMaxCountsChanged method	243
4.3.13.2	IBMDSwitcherMediaPoolCallback::FrameTotalForClipsChanged method	243
4.3.14	IBMDSwitcherMediaPool Interface	244
4.3.14.1	IBMDSwitcherMediaPool::GetStills method	244
4.3.14.2	IBMDSwitcherMediaPool::GetClip method	245
4.3.14.3	IBMDSwitcherMediaPool::GetClipCount method	245
4.3.14.4	IBMDSwitcherMediaPool::CreateFrame method	246
4.3.14.5	IBMDSwitcherMediaPool::CreateAudio method	246
4.3.14.6	IBMDSwitcherMediaPool::GetFrameTotalForClips method	247
4.3.14.7	IBMDSwitcherMediaPool::GetClipMaxFrameCounts method	247
4.3.14.8	IBMDSwitcherMediaPool::Clear method	248
4.3.14.9	IBMDSwitcherMediaPool::SetClipMaxFrameCounts method	248
4.3.14.10	IBMDSwitcherMediaPool::DoesVideoModeChangeClearMediaPool method	249
4.3.14.11	IBMDSwitcherMediaPool::AddCallback method	249
4.3.14.12	IBMDSwitcherMediaPool::RemoveCallback method	250

Section 5 — Keyers 251

5.1	Key Data Types	251
5.1.1	Key Type	251
5.1.2	Fly Key Frames	251
5.1.3	Border Bevel Options	252
5.1.4	Key Event Type	252
5.1.5	Luminance Key Parameters Event Type	252
5.1.6	Chroma Key Parameters Event Type	253
5.1.7	Pattern Key Parameters Event Type	253

5.1.8	Pattern Key Parameters Event Type	254
5.1.9	DVE Key Parameters Event Type	254
5.1.10	Fly Key Parameters Event Type	255
5.1.11	Downstream Key Event Type	256
5.2	Interface Reference	257
5.2.1	IBMDSwitcherKeyIterator Interface	257
5.2.1.1	IBMDSwitcherKeyIterator::Next method	257
5.2.2	IBMDSwitcherKey Interface	258
5.2.2.1	IBMDSwitcherKey::DoesSupportAdvancedChroma method	259
5.2.2.2	IBMDSwitcherKey::GetType method	259
5.2.2.3	IBMDSwitcherKey::SetType method	259
5.2.2.4	IBMDSwitcherKey::GetInputCut method	260
5.2.2.5	IBMDSwitcherKey::SetInputCut method	260
5.2.2.6	IBMDSwitcherKey::GetInputFill method	261
5.2.2.7	IBMDSwitcherKey::SetInputFill method	261
5.2.2.8	IBMDSwitcherKey::GetFillInputAvailabilityMask method	262
5.2.2.9	IBMDSwitcherKey::GetCutInputAvailabilityMask method	262
5.2.2.10	IBMDSwitcherKey::GetOnAir method	263
5.2.2.11	IBMDSwitcherKey::SetOnAir method	263
5.2.2.12	IBMDSwitcherKey::CanBeDVEKey method	263
5.2.2.13	IBMDSwitcherKey::GetMasked method	264
5.2.2.14	IBMDSwitcherKey::SetMasked method	264
5.2.2.15	IBMDSwitcherKey::GetMaskTop method	264
5.2.2.16	IBMDSwitcherKey::SetMaskTop method	265
5.2.2.17	IBMDSwitcherKey::GetMaskBottom method	265
5.2.2.18	IBMDSwitcherKey::SetMaskBottom method	265
5.2.2.19	IBMDSwitcherKey::GetMaskLeft method	266
5.2.2.20	IBMDSwitcherKey::SetMaskLeft method	266
5.2.2.21	IBMDSwitcherKey::GetMaskRight method	266
5.2.2.22	IBMDSwitcherKey::SetMaskRight method	267
5.2.2.23	IBMDSwitcherKey::ResetMask method	267
5.2.2.24	IBMDSwitcherKey::GetTransitionSelectionMask method	267
5.2.2.25	IBMDSwitcherKey::AddCallback method	268
5.2.2.26	IBMDSwitcherKey::RemoveCallback method	268
5.2.3	IBMDSwitcherKeyCallback Interface	269
5.2.3.1	IBMDSwitcherKeyCallback::Notify method	269
5.2.4	IBMDSwitcherKeyLumaParameters Interface	270
5.2.4.1	IBMDSwitcherKeyLumaParameters::GetPreMultiplied method	270
5.2.4.2	IBMDSwitcherKeyLumaParameters::SetPreMultiplied method	271
5.2.4.3	IBMDSwitcherKeyLumaParameters::GetClip method	271
5.2.4.4	IBMDSwitcherKeyLumaParameters::SetClip method	272
5.2.4.5	IBMDSwitcherKeyLumaParameters::GetGain method	272
5.2.4.6	IBMDSwitcherKeyLumaParameters::SetGain method	272
5.2.4.7	IBMDSwitcherKeyLumaParameters::GetInverse method	273
5.2.4.8	IBMDSwitcherKeyLumaParameters::SetInverse method	273

5.2.4.9	IBMDSwitcherKeyLumaParameters::AddCallback method	274
5.2.4.10	IBMDSwitcherKeyLumaParameters::RemoveCallback method	274
5.2.5	IBMDSwitcherKeyLumaParametersCallback Interface	275
5.2.5.1	IBMDSwitcherKeyLumaParametersCallback::Notify method	275
5.2.6	IBMDSwitcherKeyChromaParameters Interface	276
5.2.6.1	IBMDSwitcherKeyChromaParameters::GetHue method	276
5.2.6.2	IBMDSwitcherKeyChromaParameters::SetHue method	277
5.2.6.3	IBMDSwitcherKeyChromaParameters::GetGain method	277
5.2.6.4	IBMDSwitcherKeyChromaParameters::SetGain method	277
5.2.6.5	IBMDSwitcherKeyChromaParameters::GetYSuppress method	278
5.2.6.6	IBMDSwitcherKeyChromaParameters::SetYSuppress method	278
5.2.6.7	IBMDSwitcherKeyChromaParameters::GetLift method	278
5.2.6.8	IBMDSwitcherKeyChromaParameters::SetLift method	279
5.2.6.9	IBMDSwitcherKeyChromaParameters::GetNarrow method	279
5.2.6.10	IBMDSwitcherKeyChromaParameters::SetNarrow method	279
5.2.6.11	IBMDSwitcherKeyChromaParameters::AddCallback method	280
5.2.6.12	IBMDSwitcherKeyChromaParameters::RemoveCallback method	280
5.2.7	IBMDSwitcherKeyChromaParametersCallback Interface	281
5.2.7.1	IBMDSwitcherKeyChromaParametersCallback::Notify method	281
5.2.8	IBMDSwitcherKeyAdvancedChromaParameters Interface	282
5.2.8.1	IBMDSwitcherKeyAdvancedChromaParameters:: GetForegroundLevel method	283
5.2.8.2	IBMDSwitcherKeyAdvancedChromaParameters::SetForegroundLevel method	284
5.2.8.3	IBMDSwitcherKeyAdvancedChromaParameters::GetBackgroundLevel method	284
5.2.8.4	IBMDSwitcherKeyAdvancedChromaParameters::SetForegroundLevel method	284
5.2.8.5	IBMDSwitcherKeyAdvancedChromaParameters::GetKeyEdge method	285
5.2.8.6	IBMDSwitcherKeyAdvancedChromaParameters::SetKeyEdge method	285
5.2.8.7	IBMDSwitcherKeyAdvancedChromaParameters::GetSpillSuppress method	285
5.2.8.8	IBMDSwitcherKeyAdvancedChromaParameters::SetSpillSuppress method	286
5.2.8.9	IBMDSwitcherKeyAdvancedChromaParameters::GetFlareSuppress method	286
5.2.8.10	IBMDSwitcherKeyAdvancedChromaParameters::SetFlareSuppress method	286
5.2.8.11	IBMDSwitcherKeyAdvancedChromaParameters::GetBrightness method	287
5.2.8.12	IBMDSwitcherKeyAdvancedChromaParameters::SetBrightness method	287
5.2.8.13	IBMDSwitcherKeyAdvancedChromaParameters::GetContrast method	287
5.2.8.14	IBMDSwitcherKeyAdvancedChromaParameters::SetContrast method	288
5.2.8.15	IBMDSwitcherKeyAdvancedChromaParameters::GetSaturation method	288
5.2.8.16	IBMDSwitcherKeyAdvancedChromaParameters::SetSaturation method	288
5.2.8.17	IBMDSwitcherKeyAdvancedChromaParameters::GetRed method	289
5.2.8.18	IBMDSwitcherKeyAdvancedChromaParameters::SetRed method	289
5.2.8.19	IBMDSwitcherKeyAdvancedChromaParameters::GetGreen method	289
5.2.8.20	IBMDSwitcherKeyAdvancedChromaParameters::SetGreen method	290
5.2.8.21	IBMDSwitcherKeyAdvancedChromaParameters::GetBlue method	290
5.2.8.22	IBMDSwitcherKeyAdvancedChromaParameters::SetBlue method	290
5.2.8.23	IBMDSwitcherKeyAdvancedChromaParameters::GetSampling ModeEnabled method	291
5.2.8.24	IBMDSwitcherKeyAdvancedChromaParameters::SetSamplingModeEnabled method	291
5.2.8.25	IBMDSwitcherKeyAdvancedChromaParameters::GetPreviewEnabled method	292

5.2.8.26	IBMDSwitcherKeyAdvancedChromaParameters::SetPreviewEnabled method	292
5.2.8.27	IBMDSwitcherKeyAdvancedChromaParameters::GetCursorXPosition method	292
5.2.8.28	IBMDSwitcherKeyAdvancedChromaParameters::SetCursorXPosition method	293
5.2.8.29	IBMDSwitcherKeyAdvancedChromaParameters::GetCursorYPosition method	293
5.2.8.30	IBMDSwitcherKeyAdvancedChromaParameters::SetCursorYPosition method	293
5.2.8.31	IBMDSwitcherKeyAdvancedChromaParameters::GetCursorSize method	294
5.2.8.32	IBMDSwitcherKeyAdvancedChromaParameters::SetCursorSize method	294
5.2.8.33	IBMDSwitcherKeyAdvancedChromaParameters::GetSampledColor method	295
5.2.8.34	IBMDSwitcherKeyAdvancedChromaParameters::SetSampledColor method	295
5.2.8.35	IBMDSwitcherKeyAdvancedChromaParameters::ResetKeyAdjustments method	296
5.2.8.36	IBMDSwitcherKeyAdvancedChromaParameters::ResetChromaCorrection method	296
5.2.8.37	IBMDSwitcherKeyAdvancedChromaParameters::ResetColorAdjustments method	296
5.2.8.38	IBMDSwitcherKeyAdvancedChromaParameters::AddCallback method	297
5.2.8.39	IBMDSwitcherKeyAdvancedChromaParameters::RemoveCallback method	297
5.2.9	IBMDSwitcherKeyAdvancedChromaParametersCallback Interface	298
5.2.9.1	IBMDSwitcherKeyAdvancedChromaParametersCallback::Notify method	298
5.2.10	IBMDSwitcherKeyPatternParameters Interface	299
5.2.10.1	IBMDSwitcherKeyPatternParameters::GetPattern method	299
5.2.10.2	IBMDSwitcherKeyPatternParameters::SetPattern method	300
5.2.10.3	IBMDSwitcherKeyPatternParameters::GetSize method	300
5.2.10.4	IBMDSwitcherKeyPatternParameters::SetSize method	301
5.2.10.5	IBMDSwitcherKeyPatternParameters::GetSymmetry method	301
5.2.10.6	IBMDSwitcherKeyPatternParameters::SetSymmetry method	301
5.2.10.7	IBMDSwitcherKeyPatternParameters::GetSoftness method	302
5.2.10.8	IBMDSwitcherKeyPatternParameters::SetSoftness method	302
5.2.10.9	IBMDSwitcherKeyPatternParameters::GetHorizontalOffset method	302
5.2.10.10	IBMDSwitcherKeyPatternParameters::SetHorizontalOffset method	303
5.2.10.11	IBMDSwitcherKeyPatternParameters::GetVerticalOffset method	303
5.2.10.12	IBMDSwitcherKeyPatternParameters::SetVerticalOffset method	303
5.2.10.13	IBMDSwitcherKeyPatternParameters::GetInverse method	304
5.2.10.14	IBMDSwitcherKeyPatternParameters::SetInverse method	304
5.2.10.15	IBMDSwitcherKeyPatternParameters::AddCallback method	305
5.2.10.16	IBMDSwitcherKeyPatternParameters::RemoveCallback method	305
5.2.11	IBMDSwitcherKeyPatternParametersCallback Interface	306
5.2.11.1	IBMDSwitcherKeyPatternParametersCallback::Notify method	306
5.2.12	IBMDSwitcherKeyDVEParameters Interface	307
5.2.12.1	IBMDSwitcherKeyDVEParameters::GetShadow method	308
5.2.12.2	IBMDSwitcherKeyDVEParameters::SetShadow method	309
5.2.12.3	IBMDSwitcherKeyDVEParameters::GetLightSourceDirection method	309
5.2.12.4	IBMDSwitcherKeyDVEParameters::SetLightSourceDirection method	309
5.2.12.5	IBMDSwitcherKeyDVEParameters::GetLightSourceAltitude method	310
5.2.12.6	IBMDSwitcherKeyDVEParameters::SetLightSourceAltitude method	310
5.2.12.7	IBMDSwitcherKeyDVEParameters::GetBorderEnabled method	310
5.2.12.8	IBMDSwitcherKeyDVEParameters::SetBorderEnabled method	311
5.2.12.9	IBMDSwitcherKeyDVEParameters::GetBorderBevel method	311

5.2.12.10	IBMDSwitcherKeyDVEParameters::SetBorderBevel method	312
5.2.12.11	IBMDSwitcherKeyDVEParameters::GetBorderWidthIn method	312
5.2.12.12	IBMDSwitcherKeyDVEParameters::SetBorderWidthIn method	313
5.2.12.13	IBMDSwitcherKeyDVEParameters::GetBorderWidthOut method	313
5.2.12.14	IBMDSwitcherKeyDVEParameters::SetBorderWidthOut method	313
5.2.12.15	IBMDSwitcherKeyDVEParameters::GetBorderSoftnessIn method	314
5.2.12.16	IBMDSwitcherKeyDVEParameters::SetBorderSoftnessIn method	314
5.2.12.17	IBMDSwitcherKeyDVEParameters::GetBorderSoftnessOut method	314
5.2.12.18	IBMDSwitcherKeyDVEParameters::SetBorderSoftnessOut method	315
5.2.12.19	IBMDSwitcherKeyDVEParameters::GetBorderBevelSoftness method	315
5.2.12.20	IBMDSwitcherKeyDVEParameters::SetBorderBevelSoftness method	315
5.2.12.21	IBMDSwitcherKeyDVEParameters::GetBorderBevelPosition method	316
5.2.12.22	IBMDSwitcherKeyDVEParameters::SetBorderBevelPosition method	316
5.2.12.23	IBMDSwitcherKeyDVEParameters::GetBorderOpacity method	316
5.2.12.24	IBMDSwitcherKeyDVEParameters::SetBorderOpacity method	317
5.2.12.25	IBMDSwitcherKeyDVEParameters::GetBorderHue method	317
5.2.12.26	IBMDSwitcherKeyDVEParameters::SetBorderHue method	317
5.2.12.27	IBMDSwitcherKeyDVEParameters::GetBorderSaturation method	318
5.2.12.28	IBMDSwitcherKeyDVEParameters::SetBorderSaturation method	318
5.2.12.29	IBMDSwitcherKeyDVEParameters::GetBorderLuma method	318
5.2.12.30	IBMDSwitcherKeyDVEParameters::SetBorderLuma method	319
5.2.12.31	IBMDSwitcherKeyDVEParameters::GetMasked method	319
5.2.12.32	IBMDSwitcherKeyDVEParameters::SetMasked method	319
5.2.12.33	IBMDSwitcherKeyDVEParameters::GetMaskTop method	320
5.2.12.34	IBMDSwitcherKeyDVEParameters::SetMaskTop method	320
5.2.12.35	IBMDSwitcherKeyDVEParameters::GetMaskBottom method	320
5.2.12.36	IBMDSwitcherKeyDVEParameters::SetMaskBottom method	321
5.2.12.37	IBMDSwitcherKeyDVEParameters::GetMaskLeft method	321
5.2.12.38	IBMDSwitcherKeyDVEParameters::SetMaskLeft method	321
5.2.12.39	IBMDSwitcherKeyDVEParameters::GetMaskRight method	322
5.2.12.40	IBMDSwitcherKeyDVEParameters::SetMaskRight method	322
5.2.12.41	IBMDSwitcherKeyDVEParameters::ResetMask method	322
5.2.12.42	IBMDSwitcherKeyDVEParameters::AddCallback method	323
5.2.12.43	IBMDSwitcherKeyDVEParameters::RemoveCallback method	323
5.2.13	IBMDSwitcherKeyDVEParametersCallback Interface	324
5.2.13.1	IBMDSwitcherKeyDVEParametersCallback::Notify method	324
5.2.14	IBMDSwitcherKeyFlyParameters Interface	325
5.2.14.1	IBMDSwitcherKeyFlyParameters::GetFly method	326
5.2.14.2	IBMDSwitcherKeyFlyParameters::SetFly method	326
5.2.14.3	IBMDSwitcherKeyFlyParameters::GetCanFly method	326
5.2.14.4	IBMDSwitcherKeyFlyParameters::GetRate method	327
5.2.14.5	IBMDSwitcherKeyFlyParameters::SetRate method	327
5.2.14.6	IBMDSwitcherKeyFlyParameters::GetSizeX method	328
5.2.14.7	IBMDSwitcherKeyFlyParameters::SetSizeX method	328
5.2.14.8	IBMDSwitcherKeyFlyParameters::GetSizeY method	329

5.2.14.9	IBMDSwitcherKeyFlyParameters::SetSizeY method	329
5.2.14.10	IBMDSwitcherKeyFlyParameters::GetCanScaleUp method	330
5.2.14.11	IBMDSwitcherKeyFlyParameters::GetPositionX method	330
5.2.14.12	IBMDSwitcherKeyFlyParameters::SetPositionX method	331
5.2.14.13	IBMDSwitcherKeyFlyParameters::GetPositionY method	331
5.2.14.14	IBMDSwitcherKeyFlyParameters::SetPositionY method	331
5.2.14.15	IBMDSwitcherKeyFlyParameters::GetRotation method	332
5.2.14.16	IBMDSwitcherKeyFlyParameters::SetRotation method	332
5.2.14.17	IBMDSwitcherKeyFlyParameters::GetCanRotate method	332
5.2.14.18	IBMDSwitcherKeyFlyParameters::ResetRotation method	333
5.2.14.19	IBMDSwitcherKeyFlyParameters::ResetDVE method	333
5.2.14.20	IBMDSwitcherKeyFlyParameters::ResetDVEFull method	333
5.2.14.21	IBMDSwitcherKeyFlyParameters::IsKeyFrameStored method	334
5.2.14.22	IBMDSwitcherKeyFlyParameters::StoreAsKeyFrame method	334
5.2.14.23	IBMDSwitcherKeyFlyParameters::RunToKeyFrame method	335
5.2.14.24	IBMDSwitcherKeyFlyParameters::IsAtKeyFrames method	335
5.2.14.25	IBMDSwitcherKeyFlyParameters::GetKeyFrameParameters method	336
5.2.14.26	IBMDSwitcherKeyFlyParameters::IsRunning method	336
5.2.14.27	IBMDSwitcherKeyFlyParameters::AddCallback method	337
5.2.14.28	IBMDSwitcherKeyFlyParameters::RemoveCallback method	337
5.2.15	IBMDSwitcherKeyFlyParametersCallback Interface	338
5.2.15.1	IBMDSwitcherKeyFlyParametersCallback::Notify method	338
5.2.16	IBMDSwitcherKeyFlyKeyFrameParametersInterface	339
5.2.16.1	IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeX method	340
5.2.16.2	IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeX method	341
5.2.16.3	IBMDSwitcherKeyFlyKeyFrameParameters::GetSizeY method	341
5.2.16.4	IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeY method	342
5.2.16.5	IBMDSwitcherKeyFlyKeyFrameParameters::GetCanScaleUp method	342
5.2.16.6	IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionX method	343
5.2.16.7	IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionX method	343
5.2.16.8	IBMDSwitcherKeyFlyKeyFrameParameters::GetPositionY method	343
5.2.16.9	IBMDSwitcherKeyFlyKeyFrameParameters::SetPositionY method	344
5.2.16.10	IBMDSwitcherKeyFlyKeyFrameParameters::GetRotation method	344
5.2.16.11	IBMDSwitcherKeyFlyKeyFrameParameters::SetRotation method	344
5.2.16.12	IBMDSwitcherKeyFlyKeyFrameParameters::GetCanRotate method	345
5.2.16.13	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthOut method	345
5.2.16.14	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthOut method	345
5.2.16.15	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderWidthIn method	346
5.2.16.16	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderWidthIn method	346
5.2.16.17	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessOut method	346
5.2.16.18	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessOut method	347
5.2.16.19	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSoftnessIn method	347
5.2.16.20	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSoftnessIn method	347
5.2.16.21	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelSoftness method	348
5.2.16.22	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelSoftness method	348

5.2.16.23	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderBevelPosition method	348
5.2.16.24	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderBevelPosition method	349
5.2.16.25	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderOpacity method	349
5.2.16.26	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderOpacity method	349
5.2.16.27	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderHue method	350
5.2.16.28	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderHue method	350
5.2.16.29	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderSaturation method	350
5.2.16.30	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderSaturation method	351
5.2.16.31	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLuma method	351
5.2.16.32	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLuma method	351
5.2.16.33	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceDirection method	352
5.2.16.34	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLightSourceDirection method	352
5.2.16.35	IBMDSwitcherKeyFlyKeyFrameParameters::GetBorderLightSourceAltitude method	353
5.2.16.36	IBMDSwitcherKeyFlyKeyFrameParameters::SetBorderLighSourceAltitude method	353
5.2.16.37	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskTop method	354
5.2.16.38	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskTop method	354
5.2.16.39	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskBottom method	354
5.2.16.40	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskBottom method	355
5.2.16.41	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskLeft method	355
5.2.16.42	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskLeft method	355
5.2.16.43	IBMDSwitcherKeyFlyKeyFrameParameters::GetMaskRight method	356
5.2.16.44	IBMDSwitcherKeyFlyKeyFrameParameters::SetMaskRight method	356
5.2.16.45	IBMDSwitcherKeyFlyKeyFrameParameters::AddCallback method	357
5.2.16.46	IBMDSwitcherKeyFlyKeyFrameParameters::RemoveCallback method	357
5.2.17	IBMDSwitcherKeyFlyKeyFrameParametersCallback Interface	358
5.2.17.1	IBMDSwitcherKeyFlyKeyFrameParametersCallback::Notify method	358
5.2.18	IBMDSwitcherDownstreamKeyIterator Interface	359
5.2.18.1	IBMDSwitcherDownstreamKeyIterator::Next method	359
5.2.19	IBMDSwitcherDownstreamKey Interface	360
5.2.19.1	IBMDSwitcherDownstreamKey::GetInputCut method	361
5.2.19.2	IBMDSwitcherDownstreamKey::SetInputCut method	362
5.2.19.3	IBMDSwitcherDownstreamKey::GetInputFill method	362
5.2.19.4	IBMDSwitcherDownstreamKey::SetInputFill method	363
5.2.19.5	IBMDSwitcherDownstreamKey::GetFillInputAvailabilityMask method	363
5.2.19.6	IBMDSwitcherDownstreamKey::GetCutInputAvailabilityMask method	364
5.2.19.7	IBMDSwitcherDownstreamKey::GetTie method	364
5.2.19.8	IBMDSwitcherDownstreamKey::SetTie method	365
5.2.19.9	IBMDSwitcherDownstreamKey::GetRate method	365
5.2.19.10	IBMDSwitcherDownstreamKey::SetRate method	366
5.2.19.11	IBMDSwitcherDownstreamKey::GetOnAir method	366
5.2.19.12	IBMDSwitcherDownstreamKey::SetOnAir method	367
5.2.19.13	IBMDSwitcherDownstreamKey::PerformAutoTransition method	367
5.2.19.14	IBMDSwitcherDownstreamKey::PerformAutoTransitionInDirection method	367
5.2.19.15	IBMDSwitcherDownstreamKey::IsTransitioning method	368
5.2.19.16	IBMDSwitcherDownstreamKey::IsAutoTransitioning method	368

5.2.19.17	IBMDSwitcherDownstreamKey::IsTransitionTowardsOnAir method	368
5.2.19.18	IBMDSwitcherDownstreamKey::GetFramesRemaining method	369
5.2.19.19	IBMDSwitcherDownstreamKey::GetPreMultiplied method	369
5.2.19.20	IBMDSwitcherDownstreamKey::SetPreMultiplied method	369
5.2.19.21	IBMDSwitcherDownstreamKey::GetClip method	370
5.2.19.22	IBMDSwitcherDownstreamKey::SetClip method	370
5.2.19.23	IBMDSwitcherDownstreamKey::GetGain method	370
5.2.19.24	IBMDSwitcherDownstreamKey::SetGain method	371
5.2.19.25	IBMDSwitcherDownstreamKey::GetInverse method	371
5.2.19.26	IBMDSwitcherDownstreamKey::SetInverse method	371
5.2.19.27	IBMDSwitcherDownstreamKey::GetMasked method	372
5.2.19.28	IBMDSwitcherDownstreamKey::SetMasked method	372
5.2.19.29	IBMDSwitcherDownstreamKey::GetMaskTop method	372
5.2.19.30	IBMDSwitcherDownstreamKey::SetMaskTop method	373
5.2.19.31	IBMDSwitcherDownstreamKey::GetMaskBottom method	373
5.2.19.32	IBMDSwitcherDownstreamKey::SetMaskBottom method	373
5.2.19.33	IBMDSwitcherDownstreamKey::GetMaskLeft method	374
5.2.19.34	IBMDSwitcherDownstreamKey::SetMaskLeft method	374
5.2.19.35	IBMDSwitcherDownstreamKey::GetMaskRight method	374
5.2.19.36	IBMDSwitcherDownstreamKey::SetMaskRight method	375
5.2.19.37	IBMDSwitcherDownstreamKey::ResetMask method	375
5.2.19.38	IBMDSwitcherDownstreamKey::AddCallback method	376
5.2.19.39	IBMDSwitcherDownstreamKey::RemoveCallback method	376
5.2.20	IBMDSwitcherDownstreamKeyCallback Interface	377
5.2.20.1	IBMDSwitcherDownstreamKeyCallback::Notify	377

Section 6 — SuperSource 378

6.1	SuperSource Data Types	378
6.1.1	SuperSource Box Event Type	378
6.1.2	SuperSource Border Event Type	378
6.1.3	SuperSource Box Border Event Type	379
6.1.4	SuperSource Input Event Type	379
6.1.5	SuperSource Art Option	380
6.2	Interface Reference	380
6.2.1	IBMDSwitcherInputSuperSource Interface	380
6.2.1.1	IBMDSwitcherInputSuperSource::GetInputCut method	381
6.2.1.2	IBMDSwitcherInputSuperSource::SetInputCut method	381
6.2.1.3	IBMDSwitcherInputSuperSource::GetInputFill method	382
6.2.1.4	IBMDSwitcherInputSuperSource::SetInputFill method	382
6.2.1.5	IBMDSwitcherInputSuperSource::GetFillInputAvailabilityMask method	382
6.2.1.6	IBMDSwitcherInputSuperSource::GetCutInputAvailabilityMask method	383
6.2.1.7	IBMDSwitcherInputSuperSource::GetArtOption method	383
6.2.1.8	IBMDSwitcherInputSuperSource::SetArtOption method	384
6.2.1.9	IBMDSwitcherInputSuperSource::GetPreMultiplied method	384
6.2.1.10	IBMDSwitcherInputSuperSource::SetPreMultiplied method	384

6.2.1.11	IBMDSwitcherInputSuperSource::GetClip method	385
6.2.1.12	IBMDSwitcherInputSuperSource::SetClip method	385
6.2.1.13	IBMDSwitcherInputSuperSource::GetGain method	385
6.2.1.14	IBMDSwitcherInputSuperSource::SetGain method	386
6.2.1.15	IBMDSwitcherInputSuperSource::GetInverse method	386
6.2.1.16	IBMDSwitcherInputSuperSource::SetInverse method	386
6.2.1.17	IBMDSwitcherInputSuperSource::SupportsBorder method	387
6.2.1.18	IBMDSwitcherInputSuperSource::AddCallback method	387
6.2.1.19	IBMDSwitcherInputSuperSource::RemoveCallback method	387
6.2.1.20	IBMDSwitcherInputSuperSource::CreateIterator method	388
6.2.2	IBMDSwitcherInputSuperSourceCallback Interface	388
6.2.2.1	IBMDSwitcherInputSuperSourceCallback::Notify method	389
6.2.3	IBMDSwitcherSuperSourceBoxIterator Interface	389
6.2.3.1	IBMDSwitcherSuperSourceBoxIterator::Next method	390
6.2.4	IBMDSwitcherSuperSourceBox Interface	390
6.2.4.1	IBMDSwitcherSuperSourceBox::GetEnabled method	391
6.2.4.2	IBMDSwitcherSuperSourceBox::SetEnabled method	391
6.2.4.3	IBMDSwitcherSuperSourceBox::GetInputSource method	392
6.2.4.4	IBMDSwitcherSuperSourceBox::SetInputSource method	392
6.2.4.5	IBMDSwitcherSuperSourceBox::GetPositionX method	392
6.2.4.6	IBMDSwitcherSuperSourceBox::SetPositionX method	393
6.2.4.7	IBMDSwitcherSuperSourceBox::GetPositionY method	393
6.2.4.8	IBMDSwitcherSuperSourceBox::SetPositionY method	393
6.2.4.9	IBMDSwitcherSuperSourceBox::GetSize method	394
6.2.4.10	IBMDSwitcherSuperSourceBox::SetSize method	394
6.2.4.11	IBMDSwitcherSuperSourceBox::GetCropped method	394
6.2.4.12	IBMDSwitcherSuperSourceBox::SetCropped method	395
6.2.4.13	IBMDSwitcherSuperSourceBox::GetCropTop method	395
6.2.4.14	IBMDSwitcherSuperSourceBox::SetCropTop method	395
6.2.4.15	IBMDSwitcherSuperSourceBox::GetCropBottom method	396
6.2.4.16	IBMDSwitcherSuperSourceBox::SetCropBottom method	396
6.2.4.17	IBMDSwitcherSuperSourceBox::GetCropLeft method	396
6.2.4.18	IBMDSwitcherSuperSourceBox::SetCropLeft method	397
6.2.4.19	IBMDSwitcherSuperSourceBox::GetCropRight method	397
6.2.4.20	IBMDSwitcherSuperSourceBox::SetCropRight method	397
6.2.4.21	IBMDSwitcherSuperSourceBox::ResetCrop method	398
6.2.4.22	IBMDSwitcherSuperSourceBox::GetInputAvailabilityMask method	398
6.2.4.23	IBMDSwitcherSuperSourceBox::AddCallback method	398
6.2.4.24	IBMDSwitcherSuperSourceBox::RemoveCallback method	399
6.2.5	IBMDSwitcherSuperSourceBoxCallback Interface	399
6.2.5.1	IBMDSwitcherSuperSourceBoxCallback::Notify method	400
6.2.6	IBMDSwitcherSuperSourceBorder Interface	400
6.2.6.1	IBMDSwitcherSuperSourceBorder::GetBorderEnabled method	401
6.2.6.2	IBMDSwitcherSuperSourceBorder::SetBorderEnabled method	402
6.2.6.3	IBMDSwitcherSuperSourceBorder::GetBorderBevel method	402

6.2.6.4	IBMDSwitcherSuperSourceBorder::SetBorderBevel method	402
6.2.6.5	IBMDSwitcherSuperSourceBorder::GetBorderWidthOut method	403
6.2.6.7	IBMDSwitcherSuperSourceBorder::SetBorderWidthOut method	403
6.2.6.8	IBMDSwitcherSuperSourceBorder::GetBorderWidthIn method	403
6.2.6.9	IBMDSwitcherSuperSourceBorder::SetBorderWidthIn method	404
6.2.6.10	IBMDSwitcherSuperSourceBorder::GetBorderSoftnessOut method	404
6.2.6.11	IBMDSwitcherSuperSourceBorder::SetBorderSoftnessOut method	404
6.2.6.12	IBMDSwitcherSuperSourceBorder::GetBorderSoftnessIn method	405
6.2.6.13	IBMDSwitcherSuperSourceBorder::SetBorderSoftnessIn method	405
6.2.6.14	IBMDSwitcherSuperSourceBorder::GetBorderBevelSoftness method	405
6.2.6.15	IBMDSwitcherSuperSourceBorder::SetBorderBevelSoftness method	406
6.2.6.16	IBMDSwitcherSuperSourceBorder::GetBorderBevelPosition method	406
6.2.6.17	IBMDSwitcherSuperSourceBorder::SetBorderBevelPosition method	406
6.2.6.18	IBMDSwitcherSuperSourceBorder::GetBorderHue method	407
6.2.6.19	IBMDSwitcherSuperSourceBorder::SetBorderHue method	407
6.2.6.20	IBMDSwitcherSuperSourceBorder::GetBorderSaturation method	407
6.2.6.21	IBMDSwitcherSuperSourceBorder::SetBorderSaturation method	408
6.2.6.22	IBMDSwitcherSuperSourceBorder::GetBorderLuma method	408
6.2.6.23	IBMDSwitcherSuperSourceBorder::SetBorderLuma method	408
6.2.6.24	IBMDSwitcherSuperSourceBorder::GetBorderLightSourceDirection method	409
6.2.6.25	IBMDSwitcherSuperSourceBorder::SetBorderLightSourceDirection method	409
6.2.6.26	IBMDSwitcherSuperSourceBorder::GetBorderLightSourceAltitude method	409
6.2.6.27	IBMDSwitcherSuperSourceBorder::SetBorderLightSourceAltitude method	410
6.2.7	IBMDSwitcherSuperSourceBorderCallback Interface	410
6.2.7.1	IBMDSwitcherSuperSourceBorderCallback::Notify method	411
6.2.8	IBMDSwitcherSuperSourceBoxBorder Interface	411
6.2.8.1	IBMDSwitcherSuperSourceBoxBorder::GetBorderEnabled method	412
6.2.8.2	IBMDSwitcherSuperSourceBoxBorder::SetBorderEnabled method	412
6.2.8.3	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthOutHorizontal method	413
6.2.8.4	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthOutHorizontal method	413
6.2.8.5	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthOutVertical method	413
6.2.8.6	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthOutVertical method	414
6.2.8.7	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInLeft method	414
6.2.8.8	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInLeft method	414
6.2.8.9	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInRight method	415
6.2.8.10	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInRight method	415
6.2.8.11	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInTop method	415
6.2.8.12	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInTop method	416
6.2.8.13	IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInBottom method	416
6.2.8.14	IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInBottom method	416
6.2.8.15	IBMDSwitcherSuperSourceBoxBorder::GetBorderHue method	417
6.2.8.16	IBMDSwitcherSuperSourceBoxBorder::SetBorderHue method	417
6.2.8.17	IBMDSwitcherSuperSourceBoxBorder::GetBorderSaturation method	417
6.2.8.18	IBMDSwitcherSuperSourceBoxBorder::SetBorderSaturation method	418
6.2.8.19	IBMDSwitcherSuperSourceBoxBorder::GetBorderLuma method	418

6.2.8.20	IBMDSwitcherSuperSourceBoxBorder::SetBorderLuma method	418
6.2.8.21	IBMDSwitcherSuperSourceBoxBorder::AddCallback method	419
6.2.8.22	IBMDSwitcherSuperSourceBoxBorder::RemoveCallback method	419
6.2.9	IBMDSwitcherSuperSourceBoxBorderCallback Interface	420
6.2.9.1	IBMDSwitcherSuperSourceBoxBorderCallback::Notify method	420

Section 7 — Audio Mixing 421

7.1	Original Audio Mixing Data Types	421
7.1.1	Original Audio Mixer Event Type	421
7.1.2	Original Audio Mixer Audio Input Identifier	421
7.1.3	Original Audio Mixer Audio Output Identifier	421
7.1.4	Original Audio Mixer Audio Input Type	421
7.1.5	Original Audio Mixer Mix Option	421
7.1.6	Original Audio Mixer Audio Input Event Type	422
7.1.7	Original Audio Mixer Monitor Output Event Type	422
7.1.8	Original Audio Mixer Audio Headphone Output Event Type	422
7.2	Fairlight Audio Mixing Data Types	423
7.2.1	Fairlight Audio Mixer Event Type	423
7.2.2	Fairlight Audio Input Type	423
7.2.3	Fairlight Audio Input Configuration	423
7.2.4	Fairlight Audio Source Identifier	423
7.2.5	Fairlight Audio Source Type	423
7.2.6	Fairlight Audio Mix Option	424
7.2.7	Fairlight Audio Equalizer Band Shape	424
7.2.8	Fairlight Audio Equalizer Band Frequency Range	424
7.2.9	Fairlight Audio Analog Input Level	424
7.2.10	Fairlight Audio Analog Input Mic Power Mode	425
7.2.11	Fairlight Audio Input Event Type	425
7.2.12	Fairlight Audio Source Event Type	425
7.2.13	Fairlight Audio Equalizer Event Type	426
7.2.14	Fairlight Audio Equalizer Band Event Type	426
7.2.15	Fairlight Audio Dynamics Event Type	426
7.2.16	Fairlight Audio Limiter Event Type	426
7.2.17	Fairlight Audio Compressor Event Type	427
7.2.18	Fairlight Audio Expander Event Type	427
7.2.19	Fairlight Audio Headphone Output Event Type	428
7.2.20	Fairlight Audio Solo Event Type	428
7.2.21	Fairlight Analog Audio Input Event Type	428
7.2.22	Fairlight Audio Auxiliary Output Input Identifier	428
7.2.23	Fairlight Audio Auxiliary Output Identifier	429
7.2.24	Fairlight Audio Auxiliary Output Input Event Type	429
7.2.25	Fairlight Audio Auxiliary Output Event Type	429
7.3	Talkback Data Types	429
7.3.1	Switcher Talkback Event Type	429
7.3.2	Switcher Talkback ID Type	430

7.4	Original Audio Mixing Interface Reference	430
7.4.1	IBMDSwitcherAudioMixer Interface	430
7.4.1.1	IBMDSwitcherAudioMixer::GetProgramOutGain method	431
7.4.1.2	IBMDSwitcherAudioMixer::SetProgramOutGain method	431
7.4.1.3	IBMDSwitcherAudioMixer::GetProgramOutBalance method	431
7.4.1.4	IBMDSwitcherAudioMixer::GetProgramOutFollowFadeToBlack method	432
7.4.1.5	IBMDSwitcherAudioMixer::SetProgramOutFollowFadeToBlack method	432
7.4.1.6	IBMDSwitcherAudioMixer::SetProgramOutBalance method	433
7.4.1.7	IBMDSwitcherAudioMixer::GetAudioFollowVideoCrossfadeTransition	433
7.4.1.8	IBMDSwitcherAudioMixer::SetAudioFollowVideoCrossfadeTransition	434
7.4.1.9	IBMDSwitcherAudioMixer::SetAllLevelNotificationsEnable method	434
7.4.1.10	IBMDSwitcherAudioMixer::ResetProgramOutLevelNotificationPeaks method	435
7.4.1.11	IBMDSwitcherAudioMixer::ResetAllLevelNotificationPeaks method	435
7.4.1.12	IBMDSwitcherAudioMixer::AddCallback method	435
7.4.1.13	IBMDSwitcherAudioMixer::RemoveCallback method	436
7.4.1.14	IBMDSwitcherAudioMixer::CreateIterator method	436
7.4.2.1	IBMDSwitcherAudioMixerCallback::Notify method	437
7.4.2.2	IBMDSwitcherAudioMixerCallback::ProgramOutLevelNotification method	438
7.4.3	IBMDSwitcherAudioInputIterator Interface	438
7.4.3.1	IBMDSwitcherAudioInputIterator::Next method	439
7.4.3.2	IBMDSwitcherAudioInputIterator::GetById method	439
7.4.4	IBMDSwitcherAudioInput Interface	440
7.4.4.1	IBMDSwitcherAudioInput::GetType method	440
7.4.4.2	IBMDSwitcherAudioInput::GetCurrentExternalPortType method	441
7.4.4.3	IBMDSwitcherAudioInput::GetMixOption method	441
7.4.4.4	IBMDSwitcherAudioInput::SetMixOption method	442
7.4.4.5	IBMDSwitcherAudioInput::GetGain method	442
7.4.4.6	IBMDSwitcherAudioInput::SetGain method	443
7.4.4.7	IBMDSwitcherAudioInput::GetBalance method	443
7.4.4.8	IBMDSwitcherAudioInput::SetBalance method	444
7.4.4.9	IBMDSwitcherAudioInput::IsMixedIn method	444
7.4.4.10	IBMDSwitcherAudioInput::GetAudioInputId method	445
7.4.4.11	IBMDSwitcherAudioInput::ResetLevelNotificationPeaks method	445
7.4.4.12	IBMDSwitcherAudioInput::AddCallback method	446
7.4.4.13	IBMDSwitcherAudioInput::RemoveCallback method	446
7.4.5	IBMDSwitcherAudioInputCallback Interface	447
7.4.5.1	IBMDSwitcherAudioInputCallback::Notify method	447
7.4.5.2	IBMDSwitcherAudioInputCallback::LevelNotification method	448
7.4.6	IBMDSwitcherAudioInputXLR Interface	449
7.4.6.1	IBMDSwitcherAudioInputXLR::HasRCAToXLR method	449
7.4.6.2	IBMDSwitcherAudioInputXLR::GetRCAToXLREnabled method	450
7.4.6.3	IBMDSwitcherAudioInputXLR::SetRCAToXLREnabled method	450
7.4.6.4	IBMDSwitcherAudioInputXLR::AddCallback method	451
7.4.6.5	IBMDSwitcherAudioInputXLR::RemoveCallback method	451
7.4.7	IBMDSwitcherAudioInputXLRCallback Interface	452

7.4.7.1	IBMDSwitcherAudioInputXLRCallback::Notify method	452
7.4.8	IBMDSwitcherAudioMonitorOutputIterator Interface	453
7.4.8.1	IBMDSwitcherAudioMonitorOutputIterator::Next method	453
7.4.9	IBMDSwitcherAudioMonitorOutput Interface	454
7.4.9.1	IBMDSwitcherAudioMonitorOutput::GetMonitorEnable method	455
7.4.9.2	IBMDSwitcherAudioMonitorOutput::SetMonitorEnable method	455
7.4.9.3	IBMDSwitcherAudioMonitorOutput::GetMute method	456
7.4.9.4	IBMDSwitcherAudioMonitorOutput::SetMute method	456
7.4.9.5	IBMDSwitcherAudioMonitorOutput::GetGain method	456
7.4.9.6	IBMDSwitcherAudioMonitorOutput::SetGain method	457
7.4.9.7	IBMDSwitcherAudioMonitorOutput::GetSolo method	457
7.4.9.8	IBMDSwitcherAudioMonitorOutput::SetSolo method	457
7.4.9.9	IBMDSwitcherAudioMonitorOutput::GetSoloInput method	458
7.4.9.10	IBMDSwitcherAudioMonitorOutput::SetSoloInput method	458
7.4.9.11	IBMDSwitcherAudioMonitorOutput::GetDim method	459
7.4.9.12	IBMDSwitcherAudioMonitorOutput::SetDim method	459
7.4.9.13	IBMDSwitcherAudioMonitorOutput::GetDimLevel method	459
7.4.9.14	IBMDSwitcherAudioMonitorOutput::SetDimLevel method	460
7.4.9.15	IBMDSwitcherAudioMonitorOutput::ResetLevelNotificationPeaks method	460
7.4.9.16	IBMDSwitcherAudioMonitorOutput::AddCallback method	461
7.4.9.17	IBMDSwitcherAudioMonitorOutput::RemoveCallback method	461
7.4.10	IBMDSwitcherAudioMonitorOutputCallback Interface	462
7.4.10.1	IBMDSwitcherAudioMonitorOutputCallback::Notify method	462
7.4.10.2	IBMDSwitcherAudioMonitorOutputCallback::LevelNotification method	463
7.4.11	IBMDSwitcherAudioHeadphoneOutputIterator Interface	464
7.4.11.1	IBMDSwitcherAudioHeadphoneOutputIterator::Next method	464
7.4.12	IBMDSwitcherAudioHeadphoneOutput Interface	465
7.4.12.1	IBMDSwitcherAudioHeadphoneOutput::GetGain method	465
7.4.12.2	IBMDSwitcherAudioHeadphoneOutput::SetGain method	466
7.4.12.3	IBMDSwitcherAudioHeadphoneOutput::GetInputProgramOutGain method	466
7.4.12.4	IBMDSwitcherAudioHeadphoneOutput::SetInputProgramOutGain method	467
7.4.12.5	IBMDSwitcherAudioHeadphoneOutput::GetInputTalkbackGain method	467
7.4.12.6	IBMDSwitcherAudioHeadphoneOutput::SetInputTalkbackGain method	467
7.4.12.7	IBMDSwitcherAudioHeadphoneOutput::GetInputSidetoneGain method	468
7.4.12.8	IBMDSwitcherAudioHeadphoneOutput::SetInputSidetoneGain method	468
7.4.12.9	IBMDSwitcherAudioHeadphoneOutput::AddCallback method	469
7.4.12.10	IBMDSwitcherAudioHeadphoneOutput::RemoveCallback method	469
7.4.13	IBMDSwitcherAudioHeadphoneOutputCallback Interface	470
7.4.13.1	IBMDSwitcherAudioHeadphoneOutputCallback::Notify method	470
7.5	Fairlight Audio Mixing Interface Reference	471
7.5.1	IBMDSwitcherFairlightAudioMixer Interface	471
7.5.1.1	IBMDSwitcherFairlightAudioMixer::GetMasterOutEffect method	472
7.5.1.2	IBMDSwitcherFairlightAudioMixer::GetMasterOutFaderGain method	472
7.5.1.3	IBMDSwitcherFairlightAudioMixer::SetMasterOutFaderGain method	473
7.5.1.4	IBMDSwitcherFairlightAudioMixer::DoesSupportMicTalkbackGain method	473

7.5.1.5	IBMDSwitcherFairlightAudioMixer::GetMicTalkbackGain method	473
7.5.1.6	IBMDSwitcherFairlightAudioMixer::SetMicTalkbackGain method	474
7.5.1.7	IBMDSwitcherFairlightAudioMixer::GetMasterOutFollowFadeToBlack method	474
7.5.1.8	IBMDSwitcherFairlightAudioMixer::SetMasterOutFollowFadeToBlack method	475
7.5.1.9	IBMDSwitcherFairlightAudioMixer::GetAudioFollowVideoCrossfadeTransition method	475
7.5.1.10	IBMDSwitcherFairlightAudioMixer::SetAudioFollowVideoCrossfadeTransition method	476
7.5.1.11	IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled method	476
7.5.1.12	IBMDSwitcherFairlightAudioMixer::ResetMasterOutPeakLevels method	477
7.5.1.13	IBMDSwitcherFairlightAudioMixer::ResetAllPeakLevels method	477
7.5.1.14	IBMDSwitcherFairlightAudioMixer::CreateIterator method	478
7.5.1.15	IBMDSwitcherFairlightAudioMixer::AddCallback method	478
7.5.1.16	IBMDSwitcherFairlightAudioMixer::RemoveCallback method	479
7.5.2	IBMDSwitcherFairlightAudioMixerCallback Interface	479
7.5.2.1	IBMDSwitcherFairlightAudioMixerCallback::Notify method	480
7.5.2.2	IBMDSwitcherFairlightAudioMixerCallback::MasterOutLevelNotification method	481
7.5.3	IBMDSwitcherFairlightAudioInputIterator Interface	482
7.5.3.1	IBMDSwitcherFairlightAudioInputIterator::Next method	482
7.5.3.2	IBMDSwitcherFairlightAudioInputIterator::GetById method	483
7.5.4	IBMDSwitcherFairlightAudioInput Interface	483
7.5.4.1	IBMDSwitcherFairlightAudioInput::GetType method	484
7.5.4.2	IBMDSwitcherFairlightAudioInput::GetCurrentExternalPortType method	484
7.5.4.3	IBMDSwitcherFairlightAudioInput::GetSupportedConfigurations method	485
7.5.4.4	IBMDSwitcherFairlightAudioInput::GetConfiguration method	485
7.5.4.5	IBMDSwitcherFairlightAudioInput::SetConfiguration method	486
7.5.4.6	IBMDSwitcherFairlightAudioInput::GetId method	486
7.5.4.7	IBMDSwitcherFairlightAudioInput::CreateIterator method	487
7.5.4.8	IBMDSwitcherFairlightAudioInput::AddCallback method	487
7.5.4.9	IBMDSwitcherFairlightAudioInput::RemoveCallback method	488
7.5.5	IBMDSwitcherFairlightAudioInputCallback Interface	488
7.5.5.1	IBMDSwitcherFairlightAudioInputCallback::Notify method	489
7.5.6	IBMDSwitcherFairlightAudioSourceIterator Interface	489
7.5.6.1	IBMDSwitcherFairlightAudioSourceIterator::Next method	490
7.5.6.2	IBMDSwitcherFairlightAudioSourceIterator::GetById method	490
7.5.7	IBMDSwitcherFairlightAudioSource Interface	491
7.5.7.1	IBMDSwitcherFairlightAudioSource::IsActive method	492
7.5.7.2	IBMDSwitcherFairlightAudioSource::GetSourceType method	492
7.5.7.3	IBMDSwitcherFairlightAudioSource::GetMaxDelayFrames method	493
7.5.7.4	IBMDSwitcherFairlightAudioSource::GetDelayFrames method	493
7.5.7.5	IBMDSwitcherFairlightAudioSource::SetDelayFrames method	493
7.5.7.6	IBMDSwitcherFairlightAudioSource::GetInputGain method	494
7.5.7.7	IBMDSwitcherFairlightAudioSource::SetInputGain method	494
7.5.7.8	IBMDSwitcherFairlightAudioSource::HasStereoSimulation method	494
7.5.7.9	IBMDSwitcherFairlightAudioSource::GetStereoSimulationIntensity method	495
7.5.7.10	IBMDSwitcherFairlightAudioSource::SetStereoSimulationIntensity method	495
7.5.7.11	IBMDSwitcherFairlightAudioSource::GetEffect method	496

7.5.7.12	IBMDSwitcherFairlightAudioSource::GetPan method	496
7.5.7.13	IBMDSwitcherFairlightAudioSource::SetPan method	497
7.5.7.14	IBMDSwitcherFairlightAudioSource::GetFaderGain method	497
7.5.7.15	IBMDSwitcherFairlightAudioSource::SetFaderGain method	497
7.5.7.16	IBMDSwitcherFairlightAudioSource::GetSupportedMixOptions method	498
7.5.7.17	IBMDSwitcherFairlightAudioSource::GetMixOption method	498
7.5.7.18	IBMDSwitcherFairlightAudioSource::SetMixOption method	498
7.5.7.19	IBMDSwitcherFairlightAudioSource::IsMixedIn method	499
7.5.7.20	IBMDSwitcherFairlightAudioSource::ResetOutputPeakLevels method	499
7.5.7.21	IBMDSwitcherFairlightAudioSource::GetId method	499
7.5.7.22	IBMDSwitcherFairlightAudioSource::AddCallback method	500
7.5.7.23	IBMDSwitcherFairlightAudioSource::RemoveCallback method	500
7.5.8	IBMDSwitcherFairlightAudioSourceCallback Interface	501
7.5.8.1	IBMDSwitcherFairlightAudioSourceCallback::Notify method	501
7.5.8.2	IBMDSwitcherFairlightAudioSourceCallback::OutputLevelNotification method	502
7.5.9	IBMDSwitcherFairlightAudioEqualizer Interface	503
7.5.9.1	IBMDSwitcherFairlightAudioEqualizer::GetEnabled method	503
7.5.9.2	IBMDSwitcherFairlightAudioEqualizer::SetEnabled method	504
7.5.9.3	IBMDSwitcherFairlightAudioEqualizer::GetGain method	504
7.5.9.4	IBMDSwitcherFairlightAudioEqualizer::SetGain method	504
7.5.9.5	IBMDSwitcherFairlightAudioEqualizer::Reset method	505
7.5.9.6	IBMDSwitcherFairlightAudioEqualizer::CreateIterator method	505
7.5.9.7	IBMDSwitcherFairlightAudioEqualizer::AddCallback method	506
7.5.9.8	IBMDSwitcherFairlightAudioEqualizer::RemoveCallback method	506
7.5.10	IBMDSwitcherFairlightAudioEqualizerCallback Interface	507
7.5.10.1	IBMDSwitcherFairlightAudioEqualizerCallback::Notify method	507
7.5.11	IBMDSwitcherFairlightAudioEqualizerBandIterator Interface	508
7.5.11.1	IBMDSwitcherFairlightAudioEqualizerBandIterator::Next method	508
7.5.12	IBMDSwitcherFairlightAudioEqualizerBand Interface	509
7.5.12.1	IBMDSwitcherFairlightAudioEqualizerBand::GetEnabled method	510
7.5.12.2	IBMDSwitcherFairlightAudioEqualizerBand::SetEnabled method	510
7.5.12.3	IBMDSwitcherFairlightAudioEqualizerBand::GetSupportedShapes method	511
7.5.12.4	IBMDSwitcherFairlightAudioEqualizerBand::GetShape method	511
7.5.12.5	IBMDSwitcherFairlightAudioEqualizerBand::SetShape method	512
7.5.12.6	IBMDSwitcherFairlightAudioEqualizerBand::GetSupportedFrequencyRanges method	512
7.5.12.7	IBMDSwitcherFairlightAudioEqualizerBand::GetFrequencyRange method	513
7.5.12.8	IBMDSwitcherFairlightAudioEqualizerBand::SetFrequencyRange method	513
7.5.12.9	IBMDSwitcherFairlightAudioEqualizerBand::GetFrequencyRangeMinMax method	514
7.5.12.10	IBMDSwitcherFairlightAudioEqualizerBand::GetFrequency method	514
7.5.12.11	IBMDSwitcherFairlightAudioEqualizerBand::SetFrequency method	515
7.5.12.12	IBMDSwitcherFairlightAudioEqualizerBand::GetGain method	515
7.5.12.13	IBMDSwitcherFairlightAudioEqualizerBand::SetGain method	515
7.5.12.14	IBMDSwitcherFairlightAudioEqualizerBand::GetQFactor method	516
7.5.12.15	IBMDSwitcherFairlightAudioEqualizerBand::SetQFactor method	516
7.5.12.16	IBMDSwitcherFairlightAudioEqualizerBand::Reset method	516

7.5.12.17	IBMDSwitcherFairlightAudioEqualizerBand::AddCallback method	517
7.5.12.18	IBMDSwitcherFairlightAudioEqualizerBand::RemoveCallback method	517
7.5.13	IBMDSwitcherFairlightAudioEqualizerBandCallback Interface	518
7.5.13.1	IBMDSwitcherFairlightAudioEqualizerBandCallback::Notify method	518
7.5.14	IBMDSwitcherFairlightAudioDynamicsProcessor Interface	519
7.5.14.1	IBMDSwitcherFairlightAudioDynamicsProcessor::GetProcessor method	519
7.5.14.2	IBMDSwitcherFairlightAudioDynamicsProcessor::GetMakeupGain method	520
7.5.14.3	IBMDSwitcherFairlightAudioDynamicsProcessor::SetMakeupGain method	520
7.5.14.4	IBMDSwitcherFairlightAudioDynamicsProcessor::Reset method	520
7.5.14.5	IBMDSwitcherFairlightAudioDynamicsProcessor::ResetInput PeakLevels method	521
7.5.14.6	IBMDSwitcherFairlightAudioDynamicsProcessor::ResetOutput PeakLevels method	521
7.5.14.7	IBMDSwitcherFairlightAudioDynamicsProcessor::AddCallback method	522
7.5.14.8	IBMDSwitcherFairlightAudioDynamicsProcessor::RemoveCallback method	522
7.5.15	IBMDSwitcherFairlightAudioDynamicsProcessorCallback Interface	523
7.5.15.1	IBMDSwitcherFairlightAudioDynamicsProcessorCallback::Notify method	523
7.5.15.2	IBMDSwitcherFairlightAudioDynamicsProcessorCallback::InputLevel Notification method	524
7.5.15.3	IBMDSwitcherFairlightAudioDynamicsProcessorCallback::OutputLevel Notification method	525
7.5.16	IBMDSwitcherFairlightAudioLimiter Interface	525
7.5.16.1	IBMDSwitcherFairlightAudioLimiter::GetEnabled method	526
7.5.16.2	IBMDSwitcherFairlightAudioLimiter::SetEnabled method	526
7.5.16.3	IBMDSwitcherFairlightAudioLimiter::GetThreshold method	527
7.5.16.4	IBMDSwitcherFairlightAudioLimiter::SetThreshold method	527
7.5.16.5	IBMDSwitcherFairlightAudioLimiter::GetAttack method	527
7.5.16.6	IBMDSwitcherFairlightAudioLimiter::SetAttack method	528
7.5.16.7	IBMDSwitcherFairlightAudioLimiter::GetHold method	528
7.5.16.8	IBMDSwitcherFairlightAudioLimiter::SetHold method	528
7.5.16.9	IBMDSwitcherFairlightAudioLimiter::GetRelease method	529
7.5.16.10	IBMDSwitcherFairlightAudioLimiter::SetRelease method	529
7.5.16.11	IBMDSwitcherFairlightAudioLimiter::Reset method	529
7.5.16.12	IBMDSwitcherFairlightAudioLimiter::AddCallback method	530
7.5.16.13	IBMDSwitcherFairlightAudioLimiter::RemoveCallback method	530
7.5.17	IBMDSwitcherFairlightAudioLimiterCallback Interface	531
7.5.17.1	IBMDSwitcherFairlightAudioLimiterCallback::Notify method	531
7.5.17.2	IBMDSwitcherFairlightAudioLimiterCallback::GainReductionLevelNotification method	532
7.5.18	IBMDSwitcherFairlightAudioCompressor Interface	532
7.5.18.1	IBMDSwitcherFairlightAudioCompressor::GetEnabled method	533
7.5.18.2	IBMDSwitcherFairlightAudioCompressor::SetEnabled method	533
7.5.18.3	IBMDSwitcherFairlightAudioCompressor::GetThreshold method	534
7.5.18.4	IBMDSwitcherFairlightAudioCompressor::SetThreshold method	534
7.5.18.5	IBMDSwitcherFairlightAudioCompressor::GetAttack method	534
7.5.18.6	IBMDSwitcherFairlightAudioCompressor::SetAttack method	535
7.5.18.7	IBMDSwitcherFairlightAudioCompressor::GetHold method	535
7.5.18.8	IBMDSwitcherFairlightAudioCompressor::SetHold method	535
7.5.18.9	IBMDSwitcherFairlightAudioCompressor::GetRelease method	536
7.5.18.10	IBMDSwitcherFairlightAudioCompressor::SetRelease method	536

7.5.18.11	IBMDSwitcherFairlightAudioCompressor::Reset method	536
7.5.18.12	IBMDSwitcherFairlightAudioCompressor::AddCallback method	537
7.5.18.13	IBMDSwitcherFairlightAudioCompressor::RemoveCallback method	537
7.5.19	IBMDSwitcherFairlightAudioCompressorCallback Interface	538
7.5.19.1	IBMDSwitcherFairlightAudioCompressorCallback::Notify method	538
7.5.19.2	IBMDSwitcherFairlightAudioCompressorCallback::GainReductionLevel Notification method	539
7.5.20	IBMDSwitcherFairlightAudioExpander Interface	539
7.5.20.1	IBMDSwitcherFairlightAudioExpander::GetEnabled method	540
7.5.20.2	IBMDSwitcherFairlightAudioExpander::SetEnabled method	541
7.5.20.3	IBMDSwitcherFairlightAudioExpander::GetGateMode method	541
7.5.20.4	IBMDSwitcherFairlightAudioExpander::SetGateMode method	541
7.5.20.5	IBMDSwitcherFairlightAudioExpander::GetThreshold method	542
7.5.20.6	IBMDSwitcherFairlightAudioExpander::SetThreshold method	542
7.5.20.7	IBMDSwitcherFairlightAudioExpander::GetRange method	542
7.5.20.8	IBMDSwitcherFairlightAudioExpander::SetRange method	543
7.5.20.9	IBMDSwitcherFairlightAudioExpander::GetRatio method	543
7.5.20.10	IBMDSwitcherFairlightAudioExpander::SetRatio method	543
7.5.20.11	IBMDSwitcherFairlightAudioExpander::GetAttack method	544
7.5.20.12	IBMDSwitcherFairlightAudioExpander::SetAttack method	544
7.5.20.13	IBMDSwitcherFairlightAudioExpander::GetHold method	544
7.5.20.14	IBMDSwitcherFairlightAudioExpander::SetHold method	545
7.5.20.15	IBMDSwitcherFairlightAudioExpander::GetRelease method	545
7.5.20.16	IBMDSwitcherFairlightAudioExpander::SetRelease method	545
7.5.20.17	IBMDSwitcherFairlightAudioExpander::Reset method	546
7.5.20.18	IBMDSwitcherFairlightAudioExpander::AddCallback method	546
7.5.20.19	IBMDSwitcherFairlightAudioExpander::RemoveCallback method	547
7.5.21	IBMDSwitcherFairlightAudioExpanderCallback Interface	547
7.5.21.1	IBMDSwitcherFairlightAudioExpanderCallback::Notify method	548
7.5.21.2	IBMDSwitcherFairlightAudioExpanderCallback::GainReductionLevel Notification method	548
7.5.22	IBMDSwitcherFairlightAudioHeadphoneOutputIterator Interface	549
7.5.22.1	IBMDSwitcherFairlightAudioHeadphoneOutputIterator::Next method	549
7.5.23	IBMDSwitcherFairlightAudioHeadphoneOutput Interface	550
7.5.23.1	IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportSolo method	551
7.5.23.2	IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportMute method	551
7.5.23.3	IBMDSwitcherFairlightAudioHeadphoneOutput::GetGain method	552
7.5.23.4	IBMDSwitcherFairlightAudioHeadphoneOutput::SetGain method	552
7.5.23.5	IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputMasterOutGain method	553
7.5.23.6	IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputMasterOutGain method	553
7.5.23.7	IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputMasterOutMute method	554
7.5.23.8	IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputMasterOutMute method	554
7.5.23.9	IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportTalkback method	555
7.5.23.10	IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputTalkbackGain method	555
7.5.23.11	IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputTalkbackGain method	556
7.5.23.12	IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputTalkbackMute method	556
7.5.23.13	IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputTalkbackMute method	557

7.5.23.14	IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportSidetone method	557
7.5.23.15	IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputSidetoneGain method	558
7.5.23.16	IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputSidetoneGain method	558
7.5.23.17	IBMDSwitcherFairlightAudioHeadphoneOutput::AddCallback method	559
7.5.23.18	IBMDSwitcherFairlightAudioHeadphoneOutput::RemoveCallback method	559
7.5.24	IBMDSwitcherFairlightAudioHeadphoneOutputCallback Interface	560
7.5.24.1	IBMDSwitcherFairlightAudioHeadphoneOutputCallback::Notify method	560
7.5.25	IBMDSwitcherFairlightAnalogAudioInput Interface	561
7.5.25.1	IBMDSwitcherFairlightAnalogAudioInput::GetSupportedInputLevels method	561
7.5.25.2	IBMDSwitcherFairlightAnalogAudioInput::GetInputLevel method	562
7.5.25.3	IBMDSwitcherFairlightAnalogAudioInput::SetInputLevel method	562
7.5.25.4	IBMDSwitcherFairlightAnalogAudioInput::GetSupportedMicPowerModes method	563
7.5.25.5	IBMDSwitcherFairlightAnalogAudioInput::GetMicPowerMode method	563
7.5.25.6	IBMDSwitcherFairlightAnalogAudioInput::SetMicPowerMode method	564
7.5.25.7	IBMDSwitcherFairlightAnalogAudioInput::AddCallback method	564
7.5.25.8	IBMDSwitcherFairlightAnalogAudioInput::RemoveCallback method	565
7.5.26	IBMDSwitcherFairlightAnalogAudioInputCallback Interface	565
7.5.26.1	IBMDSwitcherFairlightAnalogAudioInputCallback::Notify method	566
7.5.27	IBMDSwitcherFairlightAudioSolo Interface	566
7.5.27.1	IBMDSwitcherFairlightAudioSolo::GetSolo method	567
7.5.27.2	IBMDSwitcherFairlightAudioSolo::SetSolo method	567
7.5.27.3	IBMDSwitcherFairlightAudioSolo::GetSoloInput method	568
7.5.27.4	IBMDSwitcherFairlightAudioSolo::SetSoloInput method	568
7.5.27.5	IBMDSwitcherFairlightAudioSolo::AddCallback method	569
7.5.27.6	IBMDSwitcherFairlightAudioSolo::RemoveCallback method	569
7.5.28	IBMDSwitcherFairlightAudioSoloCallback Interface	570
7.5.28.1	IBMDSwitcherFairlightAudioSoloCallback::Notify method	570
7.5.29	IBMDSwitcherFairlightAudioAuxOutputIterator Interface	571
7.5.29.1	IBMDSwitcherFairlightAudioAuxOutputIterator::Next method	571
7.5.30	IBMDSwitcherFairlightAudioAuxOutput Interface	572
7.5.30.1	IBMDSwitcherFairlightAudioAuxOutput::GetOutputId method	572
7.5.30.2	IBMDSwitcherFairlightAudioAuxOutput::DoesSupportMute method	573
7.5.30.3	IBMDSwitcherFairlightAudioAuxOutput::DoesSupportDim method	573
7.5.30.4	IBMDSwitcherFairlightAudioAuxOutput::GetGain method	573
7.5.30.5	IBMDSwitcherFairlightAudioAuxOutput::SetGain method	574
7.5.30.6	IBMDSwitcherFairlightAudioAuxOutput::GetDimOn method	574
7.5.30.7	IBMDSwitcherFairlightAudioAuxOutput::SetDimOn method	574
7.5.30.8	IBMDSwitcherFairlightAudioAuxOutput::GetDimLevel method	575
7.5.30.9	IBMDSwitcherFairlightAudioAuxOutput::SetDimLevel method	575
7.5.30.10	IBMDSwitcherFairlightAudioAuxOutput::GetMute method	575
7.5.30.11	IBMDSwitcherFairlightAudioAuxOutput::SetMute method	576
7.5.30.12	IBMDSwitcherFairlightAudioAuxOutput::CreateIterator method	576
7.5.30.13	IBMDSwitcherFairlightAudioAuxOutput::AddCallback method	577
7.5.30.14	IBMDSwitcherFairlightAudioAuxOutput::RemoveCallback method	577
7.5.31	IBMDSwitcherFairlightAudioAuxOutputCallback Interface	578

7.5.31.1	IBMDSwitcherFairlightAudioAuxOutputCallback::Notify method	578
7.5.32	IBMDSwitcherFairlightAudioAuxOutputInputIterator Interface	579
7.5.32.1	IBMDSwitcherFairlightAudioAuxOutputInputIterator::Next method	579
7.5.33	IBMDSwitcherFairlightAudioAuxOutputInput Interface	580
7.5.33.1	IBMDSwitcherFairlightAudioAuxOutputInput::GetInputId method	580
7.5.33.2	IBMDSwitcherFairlightAudioAuxOutputInput::DoesSupportMute method	581
7.5.33.3	IBMDSwitcherFairlightAudioAuxOutputInput::GetGain method	581
7.5.33.4	IBMDSwitcherFairlightAudioAuxOutputInput::SetGain method	581
7.5.33.5	IBMDSwitcherFairlightAudioAuxOutputInput::GetMute method	582
7.5.33.6	IBMDSwitcherFairlightAudioAuxOutputInput::SetMute method	582
7.5.33.7	IBMDSwitcherFairlightAudioAuxOutputInput::AddCallback method	583
7.5.33.8	IBMDSwitcherFairlightAudioAuxOutputInput::RemoveCallback method	583
7.5.34	IBMDSwitcherFairlightAudioAuxOutputInputCallback Interface	584
7.5.34.1	IBMDSwitcherFairlightAudioAuxOutputInputCallback::Notify method	584
7.6	Talkback Interface Reference	585
7.6.1	IBMDSwitcherTalkbackIterator Interface	585
7.6.1.1	IBMDSwitcherTalkbackIterator::Next method	585
7.6.1.2	IBMDSwitcherTalkbackIterator::GetById method	586
7.6.2	IBMDSwitcherTalkback Interface	586
7.6.2.1	IBMDSwitcherTalkback::GetId method	587
7.6.2.2	IBMDSwitcherTalkback::GetMuteSDI method	587
7.6.2.3	IBMDSwitcherTalkback::SetMuteSDI method	588
7.6.2.4	IBMDSwitcherTalkback::InputCanMuteSDI method	588
7.6.2.5	IBMDSwitcherTalkback::CurrentInputSupportsMuteSDI method	589
7.6.2.6	IBMDSwitcherTalkback::GetInputMuteSDI method	589
7.6.2.7	IBMDSwitcherTalkback::SetInputMuteSDI method	590
7.6.2.8	IBMDSwitcherTalkback::GetMicrophoneMuted method	590
7.6.2.9	IBMDSwitcherTalkback::SetMicrophoneMuted method	591
7.6.2.10	IBMDSwitcherTalkback::CanMuteMicrophone method	591
7.6.2.11	IBMDSwitcherTalkback::AddCallback method	592
7.6.2.12	IBMDSwitcherTalkback::RemoveCallback method	592
7.6.3	IBMDSwitcherTalkbackCallback Interface	593
7.6.3.1	IBMDSwitcherTalkbackCallback::Notify method	593

Section 8 — Camera Control 594

8.1	Camera Control Data Types	594
8.1.1	Switcher Camera Control Event Type	594
8.1.2	Switcher Camera Control Parameter Type	594
8.2	Interface Reference	595
8.2.1	Switcher Camera Control Parameter Iterator	595
8.2.1.1	IBMDSwitcherCameraControlParameterIterator::Next method	595
8.2.2	IBMDSwitcherCameraControlCallback Interface	596
8.2.2.1	IBMDSwitcherCameraControlCallback::Notify method	596
8.2.3	IBMDSwitcherCameraControl Interface	597
8.2.3.1	IBMDSwitcherCameraControl::CreateIterator method	598

8.2.3.2	IBMDSwitcherCameraControl::GetPeriodicFlushInterval method	598
8.2.3.3	IBMDSwitcherCameraControl::SetPeriodicFlushInterval method	599
8.2.3.4	IBMDSwitcherCameraControl::GetParameterInfo method	599
8.2.3.5	IBMDSwitcherCameraControl::GetParameterPeriodicFlushEnabled method	600
8.2.3.6	IBMDSwitcherCameraControl::SetParameterPeriodicFlushEnabled method	600
8.2.3.7	IBMDSwitcherCameraControl::SetFlags method	601
8.2.3.8	IBMDSwitcherCameraControl::ToggleFlags method	601
8.2.3.9	IBMDSwitcherCameraControl::GetFlags method	602
8.2.3.10	IBMDSwitcherCameraControl::SetInt8s method	602
8.2.3.11	IBMDSwitcherCameraControl::OffsetInt8s method	603
8.2.3.12	IBMDSwitcherCameraControl::GetInt8s method	603
8.2.3.13	IBMDSwitcherCameraControl::SetInt16s method	604
8.2.3.14	IBMDSwitcherCameraControl::OffsetInt16s method	604
8.2.3.15	IBMDSwitcherCameraControl::GetInt16s method	605
8.2.3.16	IBMDSwitcherCameraControl::SetInt32s method	605
8.2.3.17	IBMDSwitcherCameraControl::OffsetInt32s method	606
8.2.3.18	IBMDSwitcherCameraControl::GetInt32s method	606
8.2.3.19	IBMDSwitcherCameraControl::SetInt64s method	607
8.2.3.20	IBMDSwitcherCameraControl::OffsetInt64s method	607
8.2.3.21	IBMDSwitcherCameraControl::GetInt64s method	608
8.2.3.22	IBMDSwitcherCameraControl::OffsetFloats method	608
8.2.3.23	IBMDSwitcherCameraControl::SetFloats method	609
8.2.3.24	IBMDSwitcherCameraControl::GetFloats method	609
8.2.3.25	IBMDSwitcherCameraControl::AddCallback method	610
8.2.3.26	IBMDSwitcherCameraControl::RemoveCallback method	610

Section 9 — Macros 611

9.1	General Information	611
9.1.1	Macro Indexes and Identification	611
9.1.2	Recording a Macro	611
9.1.3	Downloading a Macro	611
9.1.4	Uploading a Macro	612
9.1.5	Unsupported Operations	612
9.2	Macro Data Types	612
9.2.1	Macro Pool Event Type	612
9.2.2	Macro Control Event Type	613
9.2.3	Macro Run Status	613
9.2.4	Macro Record Status	613
9.3	Interface Reference	613
9.3.1	IBMDSwitcherMacroPool Interface	613
9.3.1.1	IBMDSwitcherMacroPool::GetMaxCount method	614
9.3.1.2	IBMDSwitcherMacroPool::Delete method	614
9.3.1.3	IBMDSwitcherMacroPool::IsValid method	615
9.3.1.4	IBMDSwitcherMacroPool::HasUnsupportedOps method	615
9.3.1.5	IBMDSwitcherMacroPool::GetName method	616

9.3.1.6	IBMDSwitcherMacroPool::SetName method	616
9.3.1.7	IBMDSwitcherMacroPool::GetDescription method	617
9.3.1.8	IBMDSwitcherMacroPool::SetDescription method	617
9.3.1.9	IBMDSwitcherMacroPool::CreateMacro method	618
9.3.1.10	IBMDSwitcherMacroPool::Upload method	618
9.3.1.11	IBMDSwitcherMacroPool::Download method	619
9.3.1.12	IBMDSwitcherMacroPool::AddCallback method	619
9.3.1.13	IBMDSwitcherMacroPool::RemoveCallback method	620
9.3.2	IBMDSwitcherTransferMacro Interface	620
9.3.2.1	IBMDSwitcherTransferMacro::Cancel method	621
9.3.2.2	IBMDSwitcherTransferMacro::GetProgress method	621
9.3.2.3	IBMDSwitcherTransferMacro::GetMacro method	621
9.3.3	IBMDSwitcherMacro Interface	622
9.3.3.1	IBMDSwitcherMacro::GetSize method	622
9.3.3.2	IBMDSwitcherMacro::GetBytes method	622
9.3.4	IBMDSwitcherMacroPoolCallback Interface	623
9.3.4.1	IBMDSwitcherMacroPoolCallback::Notify method	623
9.3.5	IBMDSwitcherMacroControl Interface	624
9.3.5.1	IBMDSwitcherMacroControl::Run method	624
9.3.5.2	IBMDSwitcherMacroControl::GetLoop method	625
9.3.5.3	IBMDSwitcherMacroControl::SetLoop method	625
9.3.5.4	IBMDSwitcherMacroControl::ResumeRunning method	626
9.3.5.5	IBMDSwitcherMacroControl::StopRunning method	626
9.3.5.6	IBMDSwitcherMacroControl::Record method	627
9.3.5.7	IBMDSwitcherMacroControl::RecordUserWait method	627
9.3.5.8	IBMDSwitcherMacroControl::RecordPause method	628
9.3.5.9	IBMDSwitcherMacroControl::StopRecording method	628
9.3.5.10	IBMDSwitcherMacroControl::GetRunStatus method	629
9.3.5.11	IBMDSwitcherMacroControl::GetRecordStatus method	629
9.3.5.12	IBMDSwitcherMacroControl::AddCallback method	630
9.3.5.13	IBMDSwitcherMacroControl::RemoveCallback method	630
9.3.6	IBMDSwitcherMacroControlCallback Interface	631
9.3.6.1	IBMDSwitcherMacroControlCallback::Notify method	631

Section 10 — HyperDeck 632

10.1	General Information	632
10.1.1	HyperDeck Interfaces	632
10.1.2	HyperDeck Remote Control	632
10.1.3	HyperDeck Clip Cache	632
10.1.4	HyperDeck Configuration	632
10.1.5	HyperDeck Clip Cache Configuration	633
10.2	Hyperdeck Types	633
10.2.1	BMDSwitcherHyperDeckClipEventType	633
10.2.2	BMDSwitcherHyperDeckEventType	633
10.2.3	BMDSwitcherHyperDeckClipId	634

10.2.4	BMDSwitcherHyperDeckId	634
10.2.5	BMDSwitcherHyperDeckPlayerState	634
10.2.6	BMDSwitcherHyperDeckConnectionStatus	635
10.2.7	BMDSwitcherHyperDeckStorageMediaState	635
10.2.8	BMDSwitcherHyperDeckErrorType	635
10.3	Interface Reference	636
10.3.1	IBMDSwitcherHyperDeckIterator Interface	636
10.3.1.1	IBMDSwitcherHyperDeckIterator::Next method	636
10.3.1.2	IBMDSwitcherHyperDeckIterator::GetById method	637
10.3.2	IBMDSwitcherHyperDeck Interface	637
10.3.2.1	IBMDSwitcherHyperDeck::GetId method	639
10.3.2.2	IBMDSwitcherHyperDeck::GetConnectionStatus method	639
10.3.2.3	IBMDSwitcherHyperDeck::IsRemoteAccessEnabled method	640
10.3.2.4	IBMDSwitcherHyperDeck::GetStorageMediaCount method	640
10.3.2.5	IBMDSwitcherHyperDeck::GetStorageMediaState method	641
10.3.2.6	IBMDSwitcherHyperDeck::GetActiveStorageMedia method	641
10.3.2.7	IBMDSwitcherHyperDeck::SetActiveStorageMedia method	642
10.3.2.8	IBMDSwitcherHyperDeck::GetClipCount method	642
10.3.2.9	IBMDSwitcherHyperDeck::CreateIterator method	643
10.3.2.10	IBMDSwitcherHyperDeck::GetSwitcherInput method	643
10.3.2.11	IBMDSwitcherHyperDeck::SetSwitcherInput method	644
10.3.2.12	IBMDSwitcherHyperDeck::GetFrameRate method	644
10.3.2.13	IBMDSwitcherHyperDeck::IsInterlacedVideo method	645
10.3.2.14	IBMDSwitcherHyperDeck::IsDropFrameTimeCode method	645
10.3.2.15	IBMDSwitcherHyperDeck::GetPlayerState method	646
10.3.2.16	IBMDSwitcherHyperDeck::GetCurrentClip method	646
10.3.2.17	IBMDSwitcherHyperDeck::SetCurrentClip method	647
10.3.2.18	IBMDSwitcherHyperDeck::GetCurrentClipTime method	647
10.3.2.19	IBMDSwitcherHyperDeck::SetCurrentClipTime method	648
10.3.2.20	IBMDSwitcherHyperDeck::GetCurrentTimelineTime method	648
10.3.2.21	IBMDSwitcherHyperDeck::SetCurrentTimelineTime method	649
10.3.2.22	IBMDSwitcherHyperDeck::GetEstimatedRecordTimeRemaining method	649
10.3.2.23	IBMDSwitcherHyperDeck::Play method	650
10.3.2.24	IBMDSwitcherHyperDeck::Record method	650
10.3.2.25	IBMDSwitcherHyperDeck::Stop method	650
10.3.2.26	IBMDSwitcherHyperDeck::Shuttle method	651
10.3.2.27	IBMDSwitcherHyperDeck::GetShuttleSpeed method	651
10.3.2.28	IBMDSwitcherHyperDeck::Jog method	651
10.3.2.29	IBMDSwitcherHyperDeck::GetLoopedPlayback method	652
10.3.2.30	IBMDSwitcherHyperDeck::SetLoopedPlayback method	652
10.3.2.31	IBMDSwitcherHyperDeck::GetSingleClipPlayback method	653
10.3.2.32	IBMDSwitcherHyperDeck::SetSingleClipPlayback method	653
10.3.2.33	IBMDSwitcherHyperDeck::GetAutoRollOnTake method	654
10.3.2.34	IBMDSwitcherHyperDeck::SetAutoRollOnTake method	654
10.3.2.35	IBMDSwitcherHyperDeck::GetAutoRollOnTakeFrameDelay method	655

10.3.2.36	IBMDSwitcherHyperDeck::SetAutoRollOnTakeFrameDelay method	655
10.3.2.37	IBMDSwitcherHyperDeck::GetNetworkAddress method	656
10.3.2.38	IBMDSwitcherHyperDeck::SetNetworkAddress method	656
10.3.2.39	IBMDSwitcherHyperDeck::GetModelName method	657
10.3.2.40	IBMDSwitcherHyperDeck::GetStorageMediaInfo method	657
10.3.2.41	IBMDSwitcherHyperDeck::AddCallback method	658
10.3.2.42	IBMDSwitcherHyperDeck::RemoveCallback method	658
10.3.3	IBMDSwitcherHyperDeckCallback Interface	659
10.3.3.1	IBMDSwitcherHyperDeckCallback::Notify method	659
10.3.3.2	IBMDSwitcherHyperDeckCallback::NotifyError method	660
10.3.4	IBMDSwitcherHyperDeckClipIterator Interface	660
10.3.4.1	IBMDSwitcherHyperDeckClipIterator::Next method	661
10.3.4.2	IBMDSwitcherHyperDeckClipIterator::GetById method	661
10.3.5	IBMDSwitcherHyperDeckClip Interface	662
10.3.5.1	IBMDSwitcherHyperDeckClip::IsValid method	662
10.3.5.2	IBMDSwitcherHyperDeckClip::IsInfoAvailable method	663
10.3.5.3	IBMDSwitcherHyperDeckClip::GetId method	663
10.3.5.4	IBMDSwitcherHyperDeckClip::GetName method	664
10.3.5.5	IBMDSwitcherHyperDeckClip::GetDuration method	664
10.3.5.6	IBMDSwitcherHyperDeckClip::GetTimelineStart method	665
10.3.5.7	IBMDSwitcherHyperDeckClip::GetTimelineEnd method	665
10.3.5.8	IBMDSwitcherHyperDeckClip::AddCallback method	666
10.3.5.9	IBMDSwitcherHyperDeckClip::RemoveCallback method	666
10.3.6	IBMDSwitcherHyperDeckClipCallback Interface	667
10.3.6.1	IBMDSwitcherHyperDeckClipCallback::Notify method	667

Section 11 — Streaming 668

11.1	General Information	668
11.1.1	Video and Audio Encoding	668
11.2	Streaming Data Types	668
11.2.1	Streaming State	668
11.2.2	Streaming Error	668
11.2.3	Streaming Event Type	669
11.2.4	Video Codec Type	669
11.2.5	Streaming Down Convert Mode	669
11.3	Interface Reference	670
11.3.1	IBMDSwitcherStreamRTMP Interface	670
11.3.1.1	IBMDSwitcherStreamRTMP::StartStreaming method	671
11.3.1.2	IBMDSwitcherStreamRTMP::StopStreaming method	671
11.3.1.3	IBMDSwitcherStreamRTMP::IsStreaming method	672
11.3.1.4	IBMDSwitcherStreamRTMP::GetStatus method	672
11.3.1.5	IBMDSwitcherStreamRTMP::CanStreamRTMP method	673
11.3.1.6	IBMDSwitcherStreamRTMP::CanStreamRTMPS method	673
11.3.1.7	IBMDSwitcherStreamRTMP::CanStreamSRT method	673
11.3.1.8	IBMDSwitcherStreamRTMP::CanStream4K method	674

11.3.1.9	IBMDSwitcherStreamRTMP::CanDownConvert method	674
11.3.1.10	IBMDSwitcherStreamRTMP::SetServiceName method	675
11.3.1.11	IBMDSwitcherStreamRTMP::GetServiceName method	675
11.3.1.12	IBMDSwitcherStreamRTMP::SetUrl method	676
11.3.1.13	IBMDSwitcherStreamRTMP::GetUrl method	676
11.3.1.14	IBMDSwitcherStreamRTMP::SetKey method	676
11.3.1.15	IBMDSwitcherStreamRTMP::GetKey method	677
11.3.1.16	IBMDSwitcherStreamRTMP::SetVideoBitrates method	677
11.3.1.17	IBMDSwitcherStreamRTMP::GetVideoBitrates method	678
11.3.1.18	IBMDSwitcherStreamRTMP::SetAudioBitrates method	678
11.3.1.19	IBMDSwitcherStreamRTMP::GetAudioBitrates method	679
11.3.1.20	IBMDSwitcherStreamRTMP::RequestDuration method	679
11.3.1.21	IBMDSwitcherStreamRTMP::GetDuration method	680
11.3.1.22	IBMDSwitcherStreamRTMP::GetTimecode method	680
11.3.1.23	IBMDSwitcherStreamRTMP::GetEncodingBitrate method	681
11.3.1.24	IBMDSwitcherStreamRTMP::GetCacheUsed method	681
11.3.1.25	IBMDSwitcherStreamRTMP::SetDownConvertMode method	681
11.3.1.26	IBMDSwitcherStreamRTMP::GetDownConvertMode method	682
11.3.1.27	IBMDSwitcherStreamRTMP::SetAuthentication method	682
11.3.1.28	IBMDSwitcherStreamRTMP::GetAuthentication method	683
11.3.1.29	IBMDSwitcherStreamRTMP::SetLowLatency method	683
11.3.1.30	IBMDSwitcherStreamRTMP::GetLowLatency method	684
11.3.1.31	IBMDSwitcherStreamRTMP::SetSRTExtensions method	684
11.3.1.32	IBMDSwitcherStreamRTMP::GetSRTExtensions method	684
11.3.1.33	IBMDSwitcherStreamRTMP::SetProfileXml method	685
11.3.1.34	IBMDSwitcherStreamRTMP::GetProfile method	686
11.3.1.35	IBMDSwitcherStreamRTMP::AddCallback method	687
11.3.1.36	IBMDSwitcherStreamRTMP::RemoveCallback method	687
11.3.2	IBMDSwitcherStreamRTMPCallback Interface	688
11.3.2.1	IBMDSwitcherStreamRTMPCallback::Notify method	688
11.3.2.2	IBMDSwitcherStreamRTMPCallback::NotifyStatus method Interface	689

Section 12 — Recording 690

12.1	General Information	690
12.1.1	Video and Audio Encoding	690
12.1.2	Working Set of Disks	690
12.2	Recording Data Types	690
12.2.1	Recording State	690
12.2.2	Recording Error	690
12.2.3	Recording Disk Status	691
12.2.4	Record Disk Properties	691
12.2.5	Recording Event Type	691
12.2.6	Recording Disk Availability Event Type	692
12.2.7	Recording Disk Event Type	692
12.3	Interface Reference	692

12.3.1	IBMDSwitcherRecordAV Interface	692
12.3.1.1	IBMDSwitcherRecordAV::StartRecording method	693
12.3.1.2	IBMDSwitcherRecordAV::StopRecording method	693
12.3.1.3	IBMDSwitcherRecordAV::SwitchDisk method	694
12.3.1.4	IBMDSwitcherRecordAV::IsRecording method	694
12.3.1.5	IBMDSwitcherRecordAV::GetStatus method	694
12.3.1.6	IBMDSwitcherRecordAV::SetFilename method	695
12.3.1.7	IBMDSwitcherRecordAV::GetFilename method	695
12.3.1.8	IBMDSwitcherRecordAV::SetRecordInAllCameras method	696
12.3.1.9	IBMDSwitcherRecordAV::GetRecordInAllCameras method	696
12.3.1.10	IBMDSwitcherRecordAV::DoesSupportISORecording method	697
12.3.1.11	IBMDSwitcherRecordAV::SetRecordAllISOInputs method	697
12.3.1.12	IBMDSwitcherRecordAV::GetRecordAllISOInputs method	698
12.3.1.13	IBMDSwitcherRecordAV::GetWorkingSetLimit method	698
12.3.1.14	IBMDSwitcherRecordAV::SetWorkingSetDisk method	699
12.3.1.15	IBMDSwitcherRecordAV::GetWorkingSetDisk method	699
12.3.1.16	IBMDSwitcherRecordAV::GetActiveDiskIndex method	700
12.3.1.17	IBMDSwitcherRecordAV::RequestDuration method	700
12.3.1.18	IBMDSwitcherRecordAV::GetDuration method	701
12.3.1.19	IBMDSwitcherRecordAV::GetTimecode method	701
12.3.1.20	IBMDSwitcherRecordAV::GetTotalRecordingTimeAvailable method	702
12.3.1.21	IBMDSwitcherRecordAV::CreateIterator method	702
12.3.1.22	IBMDSwitcherRecordAV::AddCallback method	703
12.3.1.23	IBMDSwitcherRecordAV::RemoveCallback method	703
12.3.2	IBMDSwitcherRecordAVCallback Interface	704
12.3.2.1	IBMDSwitcherRecordAVCallback::Notify method	704
12.3.2.2	IBMDSwitcherRecordAVCallback::NotifyWorkingSetChange method	705
12.3.2.3	IBMDSwitcherRecordAVCallback::NotifyDiskAvailability method	705
12.3.2.4	IBMDSwitcherRecordAVCallback::NotifyStatus method	706
12.3.3	IBMDSwitcherRecordDiskIterator Interface	706
12.3.3.1	IBMDSwitcherRecordDiskIterator::Next method	707
12.3.3.2	IBMDSwitcherRecordDiskIterator::GetById method	707
12.3.4	IBMDSwitcherRecordDisk Interface	708
12.3.4.1	IBMDSwitcherRecordDisk::GetId method	708
12.3.4.2	IBMDSwitcherRecordDisk::GetVolumeName method	709
12.3.4.3	IBMDSwitcherRecordDisk::GetRecordingTimeAvailable method	709
12.3.4.4	IBMDSwitcherRecordDisk::GetStatus method	710
12.3.4.5	IBMDSwitcherRecordDisk::AddCallback method	710
12.3.4.6	IBMDSwitcherRecordDisk::RemoveCallback method	711
12.3.5	IBMDSwitcherRecordDiskCallback Interface	711
12.3.5.1	IBMDSwitcherRecordDiskCallback::Notify method	712

Section 13 — Display Clock 713

13.1	General Information	713
13.1.1	Availability	713

13.2	DisplayClock Data Types	713
13.2.1	DisplayClock Event Type	713
13.2.2	DisplayClock State Type	713
13.2.3	DisplayClock Mode Type	714
13.3	Interface Reference	714
13.3.1	IBMDSwitcherDisplayClock Interface	714
13.3.1.1	IBMDSwitcherDisplayClock::SetEnabled method	715
13.3.1.2	IBMDSwitcherDisplayClock::GetEnabled method	715
13.3.1.3	IBMDSwitcherDisplayClock::SetOpacity method	716
13.3.1.4	IBMDSwitcherDisplayClock::GetOpacity method	716
13.3.1.5	IBMDSwitcherDisplayClock::SetSize method	717
13.3.1.6	IBMDSwitcherDisplayClock::GetSize method	717
13.3.1.7	IBMDSwitcherDisplayClock::SetPositionX method	717
13.3.1.8	IBMDSwitcherDisplayClock::GetPositionX method	718
13.3.1.9	IBMDSwitcherDisplayClock::SetPositionY method	718
13.3.1.10	IBMDSwitcherDisplayClock::GetPositionY method	718
13.3.1.11	IBMDSwitcherDisplayClock::SetAutoHide method	719
13.3.1.12	IBMDSwitcherDisplayClock::GetAutoHide method	719
13.3.1.13	IBMDSwitcherDisplayClock::SetStartFrom method	720
13.3.1.14	IBMDSwitcherDisplayClock::SetStartFromFrames method	720
13.3.1.15	IBMDSwitcherDisplayClock::GetStartFrom method	721
13.3.1.16	IBMDSwitcherDisplayClock::SetClockMode method	721
13.3.1.17	IBMDSwitcherDisplayClock::GetClockMode method	722
13.3.1.18	IBMDSwitcherDisplayClock::GetClockState method	722
13.3.1.19	IBMDSwitcherDisplayClock::Start method	722
13.3.1.20	IBMDSwitcherDisplayClock::Stop method	723
13.3.1.21	IBMDSwitcherDisplayClock::Reset method	723
13.3.1.22	IBMDSwitcherDisplayClock::RequestTime method	723
13.3.1.23	IBMDSwitcherDisplayClock::AddCallback method	724
13.3.1.24	IBMDSwitcherDisplayClock::RemoveCallback method	724
13.3.2	IBMDSwitcherDisplayClockCallback Interface	725
13.3.2.1	IBMDSwitcherDisplayClockCallback::Notify method	725

Section 14 — Identity Information 726

14.1	IdentityInformation Data Types	726
14.1.1	Identity Information Event Type	726
14.2	Interface Reference	726
14.2.1	IBMDSwitcherIdentityInformation Interface	726
14.2.1.1	IBMDSwitcherIdentityInformation::GetUniqueld method	727
14.2.1.2	IBMDSwitcherIdentityInformation::GetIpAddress method	727
14.2.1.3	IBMDSwitcherIdentityInformation::GetMdnsName method	728
14.2.1.4	IBMDSwitcherIdentityInformation::GetDeviceName method	728
14.2.1.5	IBMDSwitcherIdentityInformation::AddCallback method	729
14.2.1.6	IBMDSwitcherIdentityInformation::RemoveCallback method	729
14.2.2	IBMDSwitcherIdentityInformationCallback Interface	730

14.2.2.1	IBMDSwitcherIdentityInformationCallback::Notify method	730
----------	--	-----

Section 15 — Audio Routing 731

15.1	General Information	731
15.1.1	Audio Channel Pairs	731
15.2	Audio Routing Types	731
15.2.1	Audio Output Identifier	731
15.2.2	Audio Routing Source Identifier	731
15.2.3	Audio Routing Output Identifier	731
15.2.4	Audio Channel Pair	731
15.2.5	Audio Internal Port Type	732
15.2.6	Audio Routing Source Event Type	732
15.2.7	Audio Routing Output Event Type	733
15.3	Interface Reference	733
15.3.1	IBMDSwitcherAudioRoutingSourceIterator Interface	733
15.3.1.1	IBMDSwitcherAudioRoutingSourceIterator::Next method	734
15.3.1.2	IBMDSwitcherAudioRoutingSourceIterator::GetById method	734
15.3.2	IBMDSwitcherAudioRoutingSource Interface	735
15.3.2.1	IBMDSwitcherAudioRoutingSource::GetId method	735
15.3.2.2	IBMDSwitcherAudioRoutingSource::GetAudioInputId method	736
15.3.2.3	IBMDSwitcherAudioRoutingSource::GetChannelPair method	736
15.3.2.4	IBMDSwitcherAudioRoutingSource::SetName method	737
15.3.2.5	IBMDSwitcherAudioRoutingSource::GetName method	737
15.3.2.6	IBMDSwitcherAudioRoutingSource::GetIsNameDefault method	738
15.3.2.7	IBMDSwitcherAudioRoutingSource::GetExternalPortType method	738
15.3.2.8	IBMDSwitcherAudioRoutingSource::GetInternalPortType method	739
15.3.2.9	IBMDSwitcherAudioRoutingSource::AddCallback method	739
15.3.2.10	IBMDSwitcherAudioRoutingSource::RemoveCallback method	740
15.3.3	IBMDSwitcherAudioRoutingSourceCallback Interface	740
15.3.3.1	IBMDSwitcherAudioRoutingSourceCallback::Notify method	741
15.3.4	IBMDSwitcherAudioRoutingOutputIterator Interface	741
15.3.4.1	IBMDSwitcherAudioRoutingOutputIterator::Next method	742
15.3.4.2	IBMDSwitcherAudioRoutingOutputIterator::GetById method	742
15.3.5	IBMDSwitcherAudioRoutingOutput Interface	743
15.3.5.1	IBMDSwitcherAudioRoutingOutput::GetId method	743
15.3.5.2	IBMDSwitcherAudioRoutingOutput::GetAudioOutputId method	744
15.3.5.3	IBMDSwitcherAudioRoutingOutput::GetChannelPair method	744
15.3.5.4	IBMDSwitcherAudioRoutingOutput::SetSource method	744
15.3.5.5	IBMDSwitcherAudioRoutingOutput::GetSource method	745
15.3.5.6	IBMDSwitcherAudioRoutingOutput::SetName method	745
15.3.5.7	IBMDSwitcherAudioRoutingOutput::GetName method	746
15.3.5.8	IBMDSwitcherAudioRoutingOutput::GetIsNameDefault method	746
15.3.5.9	IBMDSwitcherAudioRoutingOutput::GetExternalPortType method	747
15.3.5.10	IBMDSwitcherAudioRoutingOutput::GetInternalPortType method	747
15.3.5.11	IBMDSwitcherAudioRoutingOutput::AddCallback method	748

15.3.5.12	IBMDSwitcherAudioRoutingOutput::RemoveCallback method	748
15.3.6	IBMDSwitcherAudioRoutingOutputCallback Interface	749
15.3.6.1	IBMDSwitcherAudioRoutingOutputCallback::Notify method	749

Section 16 — Remote Sources 750

16.1	General Information	750
16.1.1	Remote Source Input Assignment and Activation	750
16.1.2	Local Remote Sources	750
16.1.3	External Remote Sources	750
16.1.4	Linking Remote Sources	750
16.2	Remote Source Types	751
16.2.1	Remote Source Configuration Source Event Type	751
16.2.2	Remote Source Internet Status Type	751
16.2.3	Remote Source Port Status Type	751
16.2.4	Remote Source Configuration Event Type	752
16.2.5	Remote Source Event Type	752
16.2.6	Remote Source External Event Type	752
16.3	Interface Reference	753
16.3.1	IBMDSwitcherRemoteSourceConfiguration Interface	753
16.3.1.1	IBMDSwitcherRemoteSourceConfiguration::AddExternalSource method	754
16.3.1.2	IBMDSwitcherRemoteSourceConfiguration::RemoveExternalSource method	754
16.3.1.3	IBMDSwitcherRemoteSourceConfiguration::GetDiscoverable method	755
16.3.1.4	IBMDSwitcherRemoteSourceConfiguration::SetDiscoverable method	755
16.3.1.5	IBMDSwitcherRemoteSourceConfiguration::ForceInternetProbe method	756
16.3.1.6	IBMDSwitcherRemoteSourceConfiguration::GetInternetAddress method	756
16.3.1.7	IBMDSwitcherRemoteSourceConfiguration::SetInternetAddress method	757
16.3.1.8	IBMDSwitcherRemoteSourceConfiguration::GetInternetPort method	757
16.3.1.9	IBMDSwitcherRemoteSourceConfiguration::SetInternetPort method	758
16.3.1.10	IBMDSwitcherRemoteSourceConfiguration::GetInternetStatus method	758
16.3.1.11	IBMDSwitcherRemoteSourceConfiguration::GetPortStatus method	759
16.3.1.12	IBMDSwitcherRemoteSourceConfiguration::GetMaxSources method	759
16.3.1.13	IBMDSwitcherRemoteSourceConfiguration::CreateIterator method	760
16.3.1.14	IBMDSwitcherRemoteSourceConfiguration::AddCallback method	760
16.3.1.15	IBMDSwitcherRemoteSourceConfiguration::RemoveCallback method	761
16.3.2	IBMDSwitcherRemoteSourceConfigurationCallback Interface	761
16.3.2.1	IBMDSwitcherRemoteSourceConfigurationCallback::Notify method	761
16.3.2.2	IBMDSwitcherRemoteSourceConfigurationCallback::NotifySource method	762
16.3.3	IBMDSwitcherRemoteSourceIterator Interface	762
16.3.3.1	IBMDSwitcherRemoteSourceIterator::Next method	763
16.3.4	IBMDSwitcherRemoteSource Interface	763
16.3.4.1	IBMDSwitcherRemoteSource::GetSourceId method	764
16.3.4.2	IBMDSwitcherRemoteSource::IsLinked method	764
16.3.4.3	IBMDSwitcherRemoteSource::GetLinkedInput method	765
16.3.4.4	IBMDSwitcherRemoteSource::SetLinkedInput method	765
16.3.4.5	IBMDSwitcherRemoteSource::UnlinkInput method	766

16.3.4.6	IBMDSwitcherRemoteSource::GetName method	766
16.3.4.7	IBMDSwitcherRemoteSource::GetModelName method	767
16.3.4.8	IBMDSwitcherRemoteSource::GetEnabled method	767
16.3.4.9	IBMDSwitcherRemoteSource::SetEnabled method	768
16.3.4.10	IBMDSwitcherRemoteSource::GetConnected method	768
16.3.4.11	IBMDSwitcherRemoteSource::AddCallback method	769
16.3.4.12	IBMDSwitcherRemoteSource::RemoveCallback method	769
16.3.5	IBMDSwitcherRemoteSourceCallback Interface	770
16.3.5.1	IBMDSwitcherRemoteSourceCallback::Notify method	770
16.3.6	IBMDSwitcherRemoteSourceExternal Interface	771
16.3.6.1	IBMDSwitcherRemoteSourceExternal::GetSourceId method	771
16.3.6.2	IBMDSwitcherRemoteSourceExternal::GetKey method	772
16.3.6.3	IBMDSwitcherRemoteSourceExternal::SetKey method	772
16.3.6.4	IBMDSwitcherRemoteSourceExternal::GenerateKey method	773
16.3.6.5	IBMDSwitcherRemoteSourceExternal::GetName method	773
16.3.6.6	IBMDSwitcherRemoteSourceExternal::SetName method	773
16.3.6.7	IBMDSwitcherRemoteSourceExternal::GetEnabled method	774
16.3.6.8	IBMDSwitcherRemoteSourceExternal::SetEnabled method	774
16.3.6.9	IBMDSwitcherRemoteSourceExternal::RequestStreamingXML method	775
16.3.6.10	IBMDSwitcherRemoteSourceExternal::GetStreamingXML method	775
16.3.6.11	IBMDSwitcherRemoteSourceExternal::AddCallback method	776
16.3.6.12	IBMDSwitcherRemoteSourceExternal::RemoveCallback method	776
16.3.7	IBMDSwitcherRemoteSourceExternalCallback Interface	777
16.3.7.1	IBMDSwitcherRemoteSourceExternalCallback::Notify method	777

Section 17 — VISCA PTZ Camera Control 778

17.1	General Information	778
17.1.1	RS422 Serial Port Connected Devices	778
17.1.2	IP Network Connected Devices	778
17.1.3	VISCA Camera Input Mapping	778
17.2	VISCA PTZ Camera Control Data Types	778
17.2.1	VISCA Device ID	778
17.2.2	VISCA Event Type	778
17.2.3	VISCA Device Connection Type	779
17.2.4	VISCA Device State	779
17.2.5	VISCA Device Event Type	779
17.2.6	VISCA Ping Result	779
17.3	Interface Reference	780
17.3.1	IBMDSwitcherRemoteSourceConfiguration Interface	780
17.3.1.1	IBMDSwitcherVisca::DoesSupportIP method	780
17.3.1.2	IBMDSwitcherVisca::DoesSupportSerial method	781
17.3.1.3	IBMDSwitcherVisca::MaxIPDevices method	781
17.3.1.4	IBMDSwitcherVisca::MaxSerialDevices method	781
17.3.1.5	IBMDSwitcherVisca::AddIPDevice method	782
17.3.1.6	IBMDSwitcherVisca::RemoveDevice method	782

17.3.1.7	IBMDSwitcherVisca::PingIPAddress method	783
17.3.1.8	IBMDSwitcherVisca::ScanSerialDevices method	783
17.3.1.9	IBMDSwitcherVisca::CreateIterator method	784
17.3.1.10	IBMDSwitcherVisca::AddCallback method	784
17.3.1.11	IBMDSwitcherVisca::RemoveCallback method	785
17.3.2	IBMDSwitcherViscaCallback Interface	785
17.3.2.1	IBMDSwitcherViscaCallback::Notify method	786
17.3.2.2	IBMDSwitcherViscaCallback::PingResponse method	786
17.3.3	IBMDSwitcherViscaDeviceIterator Interface	787
17.3.3.1	IBMDSwitcherViscaDeviceIterator::Next method	787
17.3.3.2	IBMDSwitcherViscaDeviceIterator::GetById method	788
17.3.4	IBMDSwitcherViscaDevice Interface	789
17.3.4.1	IBMDSwitcherViscaDevice::GetId method	789
17.3.4.2	IBMDSwitcherViscaDevice::GetConnectionType method	790
17.3.4.3	IBMDSwitcherViscaDevice::GetState method	790
17.3.4.4	IBMDSwitcherViscaDevice::SetEnabled method	791
17.3.4.5	IBMDSwitcherViscaDevice::GetEnabled method	791
17.3.4.6	IBMDSwitcherViscaDevice::SetName method	791
17.3.4.7	IBMDSwitcherViscaDevice::GetName method	792
17.3.4.8	IBMDSwitcherViscaDevice::GetSerialAddress method	792
17.3.4.9	IBMDSwitcherViscaDevice::SetURL method	793
17.3.4.10	IBMDSwitcherViscaDevice::GetURL method	793
17.3.4.11	IBMDSwitcherViscaDevice::AddCallback method	793
17.3.4.12	IBMDSwitcherViscaDevice::RemoveCallback method	794
17.3.5	IBMDSwitcherViscaDeviceCallback Interface	794
17.3.5.1	IBMDSwitcherViscaDeviceCallback::Notify method	795

Introduction

Welcome

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

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 macOS OS.

You can download the Switcher SDK and DeckLink SDK from the Blackmagic Design support center at:

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**

If you wish to ask questions outside of the software developers forum, please contact us at:

developer@blackmagicdesign.com

Section 1 — API Design

1.1 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.

1.2 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 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

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.

1.4 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.

1.5 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.

1.5.1 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.

1.5.2 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.

1.5.3 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.

1.5.4 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.

1.6 Interface Reference

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

1.6.1 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.

1.6.1.1 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.

1.6.1.2 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.

1.6.1.3 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.

1.7 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
macOS	Switchers X.Y/macOS/Include/BMDSwitcherAPI.h

1.7.1 Basic Types

boolean

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

Windows	BOOL
macOS	bool

String

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

Windows	BSTR
macOS	CFStringRef

int64_t

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

Windows	long long
macOS	int64_t

int32_t

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

Windows	int
macOS	int32_t

uint32_t

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

Windows	unsigned int
macOS	uint32_t

int16_t

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

Windows	short
macOS	int16_t

uint16_t

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

Windows	unsigned short
macOS	uint16_t

int8_t

The signed 8 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	signed char
macOS	int8_t

uint8_t

The 8 bit unsigned type is represented differently on each platform, using the most appropriate system type:

Windows	unsigned short
macOS	uint8_t

1.7.2 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:

1.7.2.1 Windows

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

```
IBMDSwitcherDiscovery* 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.

1.7.2.2 macOS OS

On macOS OS 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

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.

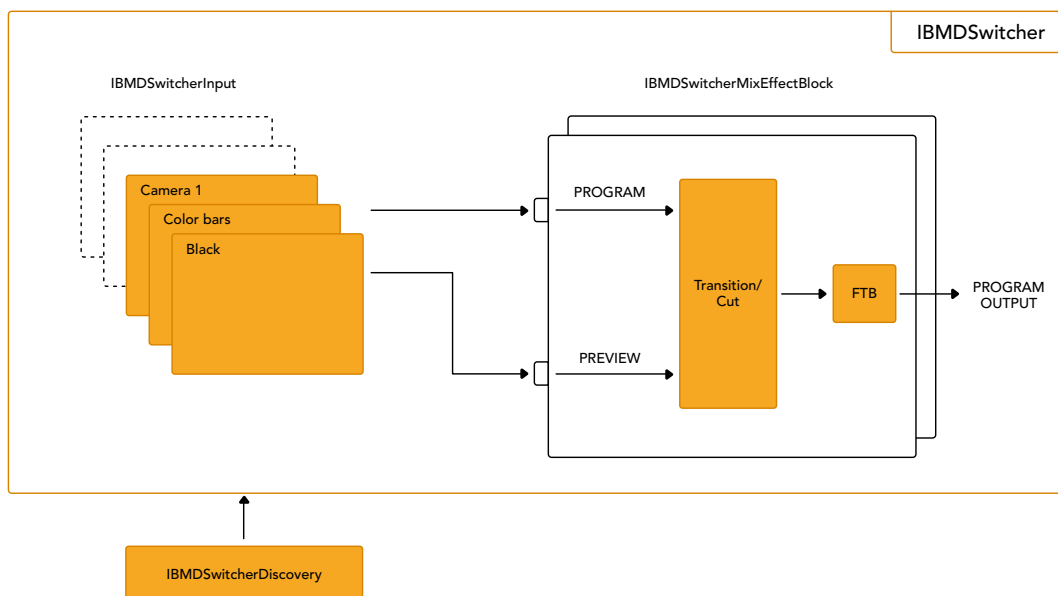
2.1 General Information

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::GetShortName` or `IBMDSwitcherInput::GetLongName`
- Use `IBMDSwitcher::CreateIterator` to get an `IBMDSwitcherMixEffectBlockIterator` object interface
- Obtain an `IBMDSwitcherMixEffectBlock` object interface using `IBMDSwitcherMixEffectBlockIterator::Next`
- Use `IBMDSwitcherMixEffectBlock::SetProgramInput` and `IBMDSwitcherMixEffectBlock::SetPreviewInput` to set 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::PerformAutoTransition`
- Install a callback using `IBMDSwitcherMixEffectBlock::AddCallback` and receive `IBMDSwitcherMixEffectBlockCallback::Notify` callbacks when `IBMDSwitcherMixEffectBlock` events occur

2.1.2 Switcher Interface Diagram



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 **IBMDSwitcherCallback**.

- **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.
- **bmdSwitcherEventType3GSDIOutputLevelChanged**
The 3GSDI output level changed.
- **bmdSwitcherEventTypeColorimetryModeChanged**
The colorimetry mode has changed.
- **bmdSwitcherEventTypeTimeCodeChanged**
The current timecode has changed. This only occurs when another event causes the currently cached timecode to be updated.
- **bmdSwitcherEventTypeTimeCodeLockedChanged**
The editable status of the timecode has changed.
- **bmdSwitcherEventTypeTimeCodeModeChanged**
The current timecode mode has changed.
- **bmdSwitcherEventTypeSuperSourceCascadeChanged**
The Supersource cascade mode has changed.
- **bmdSwitcherEventTypeAutoVideoModeChanged**
The auto video mode state has changed.
- **bmdSwitcherEventTypeAutoVideoModeDetectedChanged**
The auto video mode detection state has changed.
- **bmdSwitcherEventTypeFadeToBlackEnabledChanged**
The fade to black enabled state has changed.
- **bmdSwitcherEventTypeDskTallyOverrideEnabledChanged**
The Downstream Key Tally Override enabled state has changed.

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.

2.2.4 Switcher Video Mode

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

- `bmdSwitcherVideoMode525i5994NTSC`
525 pixels high, interlaced at 59.94Hz (NTSC).
- `bmdSwitcherVideoMode625i50PAL`
625 pixels high, interlaced at 50Hz (PAL).
- `bmdSwitcherVideoMode525i5994Anamorphic`
525 pixels high, interlaced at 59.94Hz (anamorphic 16:9 widescreen).
- `bmdSwitcherVideoMode625i50Anamorphic`
625 pixels high, interlaced at 50Hz (anamorphic 16:9 widescreen).
- `bmdSwitcherVideoMode720p50`
720 pixels high, progressively scanned at 50Hz.
- `bmdSwitcherVideoMode720p5994`
720 pixels high, progressively scanned at 59.94Hz.
- `bmdSwitcherVideoMode720p60`
720 pixels high, progressively scanned at 60Hz.
- `bmdSwitcherVideoMode1080i50`
1080 pixels high, interlaced at 50Hz.
- `bmdSwitcherVideoMode1080i5994`
1080 pixels high, interlaced at 59.94Hz.
- `bmdSwitcherVideoMode1080i60`
1080 pixels high, interlaced at 60Hz.
- `bmdSwitcherVideoMode1080p2398`
1080 pixels high, progressively scanned at 23.98Hz.
- `bmdSwitcherVideoMode1080p24`
1080 pixels high, progressively scanned at 24Hz.
- `bmdSwitcherVideoMode1080p25`
1080 pixels high, progressively scanned at 25Hz.
- `bmdSwitcherVideoMode1080p2997`
1080 pixels high, progressively scanned at 29.97Hz.
- `bmdSwitcherVideoMode1080p30`
1080 pixels high, progressively scanned at 30Hz.
- `bmdSwitcherVideoMode1080p50`
1080 pixels high, progressively scanned at 50Hz.
- `bmdSwitcherVideoMode1080p5994`
1080 pixels high, progressively scanned at 59.94Hz.
- `bmdSwitcherVideoMode1080p60`
1080 pixels high, progressively scanned at 60Hz.
- `bmdSwitcherVideoMode4KHDp2398`
2160 pixels high, progressively scanned at 23.98Hz.
- `bmdSwitcherVideoMode4KHDp24`
2160 pixels high, progressively scanned at 24Hz.
- `bmdSwitcherVideoMode4KHDp25`
2160 pixels high, progressively scanned at 25Hz.
- `bmdSwitcherVideoMode4KHDp2997`
2160 pixels high, progressively scanned at 29.97Hz.
- `bmdSwitcherVideoMode4KHDp30`
2160 pixels high, progressively scanned at 30Hz.
- `bmdSwitcherVideoMode4KHDp50`
2160 pixels high, progressively scanned at 50Hz.
- `bmdSwitcherVideoMode4KHDp5994`
2160 pixels high, progressively scanned at 59.94Hz.

- **bmdSwitcherVideoMode4KHDp60**
2160 pixels high, progressively scanned at 60Hz.
- **bmdSwitcherVideoMode8KHDp2398**
4320 pixels high, progressively scanned at 23.98Hz.
- **bmdSwitcherVideoMode8KHDp24**
4320 pixels high, progressively scanned at 24Hz.
- **bmdSwitcherVideoMode8KHDp25**
4320 pixels high, progressively scanned at 25Hz.
- **bmdSwitcherVideoMode8KHDp2997**
4320 pixels high, progressively scanned at 29.97Hz.
- **bmdSwitcherVideoMode8KHDp30**
4320 pixels high, progressively scanned at 30Hz.
- **bmdSwitcherVideoMode8KHDp50**
4320 pixels high, progressively scanned at 50Hz.
- **bmdSwitcherVideoMode8KHDp5994**
4320 pixels high, progressively scanned at 59.94Hz.
- **bmdSwitcherVideoMode8KHDp60**
4320 pixels high, progressively scanned at 60Hz.

2.2.5 Switcher Down Conversion Methods

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

- **bmdSwitcherDownConversionMethodCentreCut**
Centre cut conversion.
- **bmdSwitcherDownConversionMethodLetterbox**
Letter box conversion.
- **bmdSwitcherDownConversionMethodAnamorphic**
Anamorphic conversion.

2.2.6 Switcher Input Event Type

BMDSwitcherInputEventType enumerates the possible event types for **IBMDSwitcherInputCallback**.

- **bmdSwitcherInputEventTypeShortNameChanged**
The short name of the input changed.
- **bmdSwitcherInputEventTypeLongNameChanged**
The long name of the input changed.
- **bmdSwitcherInputEventTypesProgramTalliedChanged**
Program tallying for this input was turned on or turned off.
- **bmdSwitcherInputEventTypesPreviewTalliedChanged**
Preview tallying for this input was turned on or turned off.
- **bmdSwitcherInputEventTypeAvailableExternalPortTypesChanged**
The external port types available to this input changed.
- **bmdSwitcherInputEventTypeCurrentExternalPortTypeChanged**
The current external port type of this input changed.
- **bmdSwitcherInputEventTypeAreNamesDefaultChanged**
The long or short names changed from the default OR the long and short names changed to the default.
- **bmdSwitcherInputEventTypeInputViscaldChanged**
The VISCA device assigned to the input has changed.

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.

- **bmdSwitcherExternalPortTypeSDI**
SDI connection.
- **bmdSwitcherExternalPortTypeHDMI**
HDMI connection.
- **bmdSwitcherExternalPortTypeComponent**
Component connection.
- **bmdSwitcherExternalPortTypeComposite**
Composite connection, only available in SD video modes.
- **bmdSwitcherExternalPortTypeSVideo**
SVideo connection, only available in SD video modes.
- **bmdSwitcherExternalPortTypeXLR**
XLR audio connection.
- **bmdSwitcherExternalPortTypeAESEBU**
AES EBU audio connection.
- **bmdSwitcherExternalPortTypeRCA**
RCA audio connection.
- **bmdSwitcherExternalPortTypeTSJack**
TS audio connection.
- **bmdSwitcherExternalPortTypeMADI**
MADI audio connection.
- **bmdSwitcherExternalPortTypeTRS**
TRS audio connection.

2.2.8 Switcher Port Types

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

- **bmdSwitcherPortTypeExternal**
The port is an external port with a physical connector.
- **bmdSwitcherPortTypeBlack**
The port is a black video generator port.
- **bmdSwitcherPortTypeColorBars**
The port is a colorbars generator port.
- **bmdSwitcherPortTypeColorGenerator**
The port is a color generator port.
- **bmdSwitcherPortTypeMediaPlayerFill**
The port is a media player fill port.
- **bmdSwitcherPortTypeMediaPlayerCut**
The port is a media player cut port.
- **bmdSwitcherPortTypeSuperSource**
The port is a super source port.
- **bmdSwitcherPortTypeMixEffectBlockOutput**
The port is a mix effect block output port.
- **bmdSwitcherPortTypeAuxOutput**
The port is an auxiliary output port.
- **bmdSwitcherPortTypeMultiview**
The port is a MultiView port.
- **bmdSwitcherPortTypeExternalDirect**
The port comes from an external port with a physical connection and bypasses all switcher processing.

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.
- **bmdSwitcherInputAvailabilityMixEffectBlock2**
The input is available to be used by mix effect block 2.
- **bmdSwitcherInputAvailabilityMixEffectBlock3**
The input is available to be used by mix effect block 3.
- **bmdSwitcherInputAvailabilityAux1Output**
The input is available to be used by aux output 1.
- **bmdSwitcherInputAvailabilityAux2Output**
The input is available to be used by aux output 2.
- **bmdSwitcherInputAvailabilityWebcamOut**
The input is available to be used by webcam output.
- **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 Events

BMDSwitcherMixEffectBlockEventType enumerates the possible event types for **IBMDSwitcherMixEffectBlockCallback**.

- **bmdSwitcherMixEffectBlockEventTypeProgramInputChanged**
The program input changed.
- **bmdSwitcherMixEffectBlockEventTypePreviewInputChanged**
The preview input changed.
- **bmdSwitcherMixEffectBlockEventTypeTransitionPositionChanged**
The transition position changed.
- **bmdSwitcherMixEffectBlockEventTypeTransitionFramesRemainingChanged**
The transition frames remaining changed.
- **bmdSwitcherMixEffectBlockEventTypeInTransitionChanged**
The in transition flag changed.
- **bmdSwitcherMixEffectBlockEventTypeFadeToBlackFramesRemainingChanged**
The fade to black frames remaining changed.
- **bmdSwitcherMixEffectBlockEventTypeInFadeToBlackChanged**
The in fade to black flag changed.
- **bmdSwitcherMixEffectBlockEventTypePreviewLiveChanged**
The preview live flag changed.

- **bmdSwitcherMixEffectBlockEventTypePreviewTransitionChanged**
The preview transition changed.
- **bmdSwitcherMixEffectBlockEventTypeInputAvailabilityMaskChanged**
The input availability mask changed.
- **bmdSwitcherMixEffectBlockEventTypeFadeToBlackRateChanged**
The fade to black rate changed.
- **bmdSwitcherMixEffectBlockEventTypeFadeToBlackFullyBlackChanged**
The fade to black fully black flag changed.
- **bmdSwitcherMixEffectBlockEventTypeFadeToBlackInTransitionChanged**
The fade to black in transition flag changed.

2.2.11 Switcher Connection Errors

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

- **bmdSwitcherConnectToFailureNoResponse**
The Switcher did not respond after a connection attempt was made.
- **bmdSwitcherConnectToFailureIncompatibleFirmware**
The software on the Switcher is incompatible with the current version of the Switcher SDK.
- **bmdSwitcherConnectToFailureCorruptData**
Corrupt data was received during a connection attempt.
- **bmdSwitcherConnectToFailureStateSync**
State synchronisation failed during a connection attempt.
- **bmdSwitcherConnectToFailureStateSyncTimedOut**
State synchronisation timed out during a connection attempt.

2.2.12 Switcher MultiView Layouts

BMDSwitcherMultiViewLayout enumerates the possible layout formats for MultiView. This enumeration represents a bitset that indicates whether each quadrant of the screen displays as one large window or four small windows.

- **bmdSwitcherMultiViewLayoutProgramTop**
Program and preview reside in upper section of the screen, windows reside below.
- **bmdSwitcherMultiViewLayoutProgramBottom**
Program and preview reside in lower section of the screen, windows reside above.
- **bmdSwitcherMultiViewLayoutProgramLeft**
Program and preview reside on left hand side, windows reside on the right.
- **bmdSwitcherMultiViewLayoutProgramRight**
Program and preview reside on right hand side, windows reside on the left.
- **bmdSwitcherMultiViewLayoutTopLeftSmall**
Bitmask for the top left quadrant. If set, four small windows are shown. If not set, a single large window is shown.
- **bmdSwitcherMultiViewLayoutTopRightSmall**
Bitmask for the top right quadrant. If set, four small windows are shown. If not set, a single large window is shown.
- **bmdSwitcherMultiViewLayoutBottomLeftSmall**
Bitmask for the bottom left quadrant. If set, four small windows are shown. If not set, a single large window is shown.
- **bmdSwitcherMultiViewLayoutBottomRightSmall**
Bitmask for the bottom right quadrant. If set, four small windows are shown. If not set, a single large window is shown.

2.2.13 Switcher MultiView Safe Area Type

BMDSwitcherMultiViewSafeAreaType is a bit set that enumerates the possible safe area overlay types that can be shown in MultiView windows, used by **IBMDSwitcherMultiView**.

- **bmdSwitcherMultiViewSafeAreaTypeAspect16x9**
16:9 ratio safe area overlay.
- **bmdSwitcherMultiViewSafeAreaTypeAspect9x16**
9:16 ratio safe area overlay.

2.2.14 Switcher Serial Port Functions

BMDSwitcherSerialPortFunction enumerates the functions the serial port can perform.

- **bmdSwitcherSerialPortFunctionNone**
The serial port is not being used.
- **bmdSwitcherSerialPortFunctionPtzVisca**
The serial port is used to control a Pan-Tilt-Zoom camera using the VISCA protocol.
- **bmdSwitcherSerialPortFunctionGvg100**
If the serial port is set to this function, the switcher can be controlled by a connected GVG100 compatible editor/controller.

2.2.15 Switcher Serial Port Baud Rate

BMDSwitcherSerialPortBaudRate enumerates the possible baud rates of the serial port.

- **bmdSwitcherSerialPortBaudRate9600**
Serial port baud rate of 9600.
- **bmdSwitcherSerialPortBaudRate38400**
Serial port baud rate of 38400.

2.2.16 Switcher 3G-SDI Output Levels

BMDSwitcher3GSDIOutputLevel enumerates the possible 3G-SDI output encoding levels.

- **bmdSwitcher3GSDIOutputLevelA**
All 3G-SDI outputs use level A encoding.
- **bmdSwitcher3GSDIOutputLevelB**
All 3G-SDI outputs use level B encoding.

2.2.17 Switcher Colorimetry Mode

BMDSwitcherColorimetryMode enumerates the possible color space options for all video output.

- **bmdSwitcherColorimetryModeRec601_SDR**
Rec.601 colorspace with standard dynamic range.
- **bmdSwitcherColorimetryModeRec709_SDR**
Rec.709 colorspace with standard dynamic range.
- **bmdSwitcherColorimetryModeRec2020_SDR**
Rec.2020 colorspace with standard dynamic range.
- **bmdSwitcherColorimetryModeRec2020_HLG**
Rec.2020 colorspace with high dynamic range (HLG).
- **bmdSwitcherColorimetryModeRec2020_PQ**
Rec.2020 colorspace with high dynamic range (PQ).
- **bmdSwitcherColorimetryModeAutomatic**
Automatic colorspace and dynamic range.

2.2.18 Switcher Mix Minus Output Audio Modes

BMDSwitcherMixMinusOutputAudioMode enumerates the different audio modes for **IBMDSwitcherMixMinusOutput**.

- **bmdSwitcherMixMinusOutputAudioModeProgramOut**
The mix minus output audio is program out.
- **bmdSwitcherMixMinusOutputAudioModeMixMinus**
The mix minus output audio is program out minus the corresponding input.

2.2.19 Switcher Color Events

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

- **bmdSwitcherInputColorEventTypeHueChanged**
The hue changed.
- **bmdSwitcherInputColorEventTypeSaturationChanged**
The saturation changed.
- **bmdSwitcherInputColorEventTypeLumaChanged**
The luma changed.

2.2.20 Switcher Aux Events

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

- **bmdSwitcherInputAuxEventTypeInputSourceChanged**
The input source changed.

2.2.21 Switcher MultiView Events

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

- **bmdSwitcherMultiViewEventTypeLayoutChanged**
The layout changed.
- **bmdSwitcherMultiViewEventTypeWindowChanged**
Routing to a MultiView window has changed.
- **bmdSwitcherMultiViewEventTypeCurrentInputSupportsVuMeterChanged**
The input of a MultiView window changed to/from an input that supports VU meters from/to one that does not.
- **bmdSwitcherMultiViewEventTypeVuMeterEnabledChanged**
The enabled state of one of the VU meters changed.
- **bmdSwitcherMultiViewEventTypeVuMeterOpacityChanged**
The opacity of one of the VU meters changed.
- **bmdSwitcherMultiViewEventTypeCurrentInputSupportsSafeAreaChanged**
The input of a MultiView window changed to/from an input that supports safe area display from/to one that does not.
- **bmdSwitcherMultiViewEventTypeSafeAreaEnabledChanged**
The enabled state of the safe area overlay changed.
- **bmdSwitcherMultiViewEventTypeSafeAreaTypeChanged**
The safe area overlay in a MultiView window changed.
- **bmdSwitcherMultiViewEventTypeProgramPreviewSwappedChanged**
The positioning of the program and preview windows changed from standard to swapped or from swapped to standard.
- **bmdSwitcherMultiViewEventTypeOverlayAvailabilityChanged**
The overlay availability of a MultiView window has changed.

- **bmdSwitcherMultiViewEventTypeOverlayVisibilityChanged**
The visibility of a label or border in a MultiView window has changed.
- **bmdSwitcherMultiViewEventTypeBorderColorChanged**
The default MultiView border color has changed.

2.2.22 Switcher Serial Port Events

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

- **bmdSwitcherSerialPortEventTypeFunctionChanged**
The function of the serial port has changed.
- **bmdSwitcherSerialPortEventTypeBaudRateChanged**
The baud rate of the serial port has changed.

2.2.23 Switcher Mix Minus Output Events

BMDSwitcherMixMinusOutputEventType enumerates the possible event types for **IBMDSwitcherMixMinusOutputCallback**.

- **bmdSwitcherMixMinusOutputEventTypeAvailableAudioModesChanged**
The audio modes available to this mix minus output have changed.
- **bmdSwitcherMixMinusOutputEventTypeAudioModeChanged**
The mix minus output audio mode changed.
- **bmdSwitcherMixMinusOutputEventTypeHasMinusAudioInputIdChanged**
The mix minus output has minus audio input ID flag has changed.
- **bmdSwitcherMixMinusOutputEventTypeMinusAudioInputIdChanged**
The mix minus output minus audio input ID has changed.

2.2.24 Save And Recall Type

BMDSwitcherSaveRecallType enumerates the possible operating states for the **IBMDSwitcherSaveRecall** object interface.

- **bmdSwitcherSaveRecallTypeStartupState**
This type of operating state will be restored whenever a switcher is powered on.

2.2.25 Switcher TimeCode Mode Type

BMDSwitcherTimeCodeMode enumerates the possible modes available for running timecode.

- **bmdSwitcherTimeCodeModeFreeRun**
The timecode runs freely and typically starts at 00:00:00:00 when the switcher was powered on.
- **bmdSwitcherTimeCodeModeTimeOfDay**
The timecode runs freely and is synchronised with the current time of day.

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

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.

If a network connection cannot be established, `ConnectTo` will attempt to connect via USB if the switcher supports it.

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. Set this empty to only connect via USB.
switcherDevice	out	<code>IBMDSwitcher</code> object interface for the connected switcher.
failReason	out	Reason for connection failure as a <code>BMDSwitcherConnectToFailure</code> value.

Return Values

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

2.3.2 IBMDSwitcher Interface

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

Related Interfaces

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

Public Member Functions	
Method	Description
<code>GetProductName</code>	Get the product name of the switcher.
<code>GetVideoMode</code>	Get the current video standard applied across the switcher.
<code>SetVideoMode</code>	Set the video standard applied across the switcher.
<code>DoesSupportVideoMode</code>	Determines if a video standard is supported by the switcher.
<code>DoesVideoModeChangeRequireReconfiguration</code>	Determines if changing video standard will reconfigure the switcher.
<code>GetMethodForDownConvertedSD</code>	Get the SD conversion method applied when down converting between broadcast standards.
<code>SetMethodForDownConvertedSD</code>	Set the SD conversion method applied when down converting between broadcast standards.
<code>GetDownConvertedHDVideoMode</code>	Get the down converted HD video standard for a particular core video standard.
<code>SetDownConvertedHDVideoMode</code>	Set the down converted HD video standard for a particular core video standard.
<code>DoesSupportDownConvertedHDVideoMode</code>	Determines if a down converted HD video standard is supported by a particular core video standard.
<code>GetMultiViewVideoMode</code>	Get the MultiView video standard for a particular core video standard.
<code>SetMultiViewVideoMode</code>	Set the MultiView video standard for a particular core video standard.
<code>DoesSupportMultiViewVideoMode</code>	Determines if a MultiView video standard is supported by a particular core video standard.
<code>Get3GSDIOutputLevel</code>	Gets the current 3G-SDI output encoding level of the switcher
<code>Set3GSDIOutputLevel</code>	Sets the current 3G-SDI output encoding level of the switcher.
<code>DoesSupportColorimetrySetting</code>	Determines if changing colorimetry mode is supported by the switcher.
<code>GetColorimetryMode</code>	Get the current colorimetry mode.
<code>SetColorimetryMode</code>	Set the colorimetry mode.
<code>GetPowerStatus</code>	Gets the power status of the switcher.
<code>GetTimeCode</code>	Get the cached timecode.
<code>SetTimeCode</code>	Set the current timecode.
<code>RequestTimeCode</code>	Request the current timecode from the switcher.
<code>GetTimeCodeLocked</code>	Get the current timecode locked flag.
<code>GetTimeCodeMode</code>	Get the current timecode mode.

Public Member Functions	
Method	Description
SetTimeCodeMode	Set the current timecode mode.
GetAreOutputsConfigurable	Determines if all switcher outputs are fixed or configurable.
GetSuperSourceCascade	Get the current SuperSource cascade flag.
SetSuperSourceCascade	Set the SuperSource cascade flag.
SuspendStreaming	Suspend USB streaming.
AllowStreamingToResume	Allow USB streaming to resume.
DoesSupportAutoVideoMode	Determines if auto video mode is supported by the switcher.
GetAutoVideoMode	Get the current auto video mode state.
GetAutoVideoModeDetected	Get the current state of the input video mode detection.
SetAutoVideoMode	Set the auto video mode state.
DoesSupportFadeToBlackEnabledSetting	Determines if disabling Fade To Black control is supported by the switcher.
GetFadeToBlackEnabled	Get the current Fade To Black control state.
SetFadeToBlackEnabled	Set the Fade To Black control state.
DoesSupportDskTallyOverride	Determines if Downstream Key Tally Override is supported by the switcher.
GetDskTallyOverrideEnabled	Get the current Downstream Key Tally Override enabled flag.
SetDskTallyOverrideEnabled	Set the Downstream Key Tally Override enabled flag.
CreateIterator	Create an iterator.
AddCallback	Add a callback to receive switcher events.
RemoveCallback	Remove a callback.

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.

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.
S_FALSE	The video standard cannot be set while auto video mode is enabled.

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.

2.3.2.5 **IBMDSwitcher::DoesVideoModeChangeRequireReconfiguration** method

The **DoesVideoModeChangeRequireReconfiguration** method determines if changing to the specified video standard will cause the switcher to be reconfigured, which may result in the switcher restarting.

Syntax

```
HRESULT DoesVideoModeChangeRequireReconfiguration  
(BMDSwitcherVideoMode videoMode, boolean* required)
```

Parameters

Name	Direction	Description
videoMode	in	The video standard.
required	out	Boolean value that is true if changing to the video standard will reconfigure the switcher.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The videoMode is not a valid video standard.
E_POINTER	The required parameter is invalid.

2.3.2.6 **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.

2.3.2.7 **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.

2.3.2.8 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 <code>downConvertedHDVideoMode</code> parameter is invalid.
E_INVALIDARG	The <code>coreVideoMode</code> parameter is invalid or not supported.
E_NOTIMPL	HD down conversion is not supported.

2.3.2.9 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 <code>coreVideoMode</code> or <code>downConvertedHDVideoMode</code> parameter is invalid or not supported.
E_NOTIMPL	HD down conversion is not supported.

2.3.2.10 **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,
MDSwitcherVideoMode 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 <code>downConvertedHDVideoMode</code> is supported for the core video standard.

Return Values

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

2.3.2.11 **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 <code>multiviewVideoMode</code> parameter is invalid.
E_INVALIDARG	The <code>coreVideoMode</code> parameter is invalid or not supported.

2.3.2.12 **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.

2.3.2.13 **IBMDSwitcher::Get3GSDIOutputLevel** method

The **Get3GSDIOutputLevel** method gets the output encoding level for all 3G-SDI outputs of the switcher, on models supporting 3G-SDI video formats.

Syntax

```
HRESULT Get3GSDIOutputLevel(BMDSwitcher3GSDIOutputLevel* outputLevel)
```

Parameters

Name	Direction	Description
outputLevel	out	The current 3G-SDI output level.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The outputLevel parameter is invalid.
E_NOTIMPL	The connected switcher does not support 3G-SDI output.

2.3.2.14 **IBMDSwitcher::Set3GSDIOutputLevel** method

The **Set3GSDIOutputLevel** method sets the output encoding level for all 3G-SDI outputs of the switcher, on models supporting 3G-SDI video formats.

Syntax

```
HRESULT Set3GSDIOutputLevel(BMDSwitcher3GSDIOutputLevel outputLevel)
```

Parameters

Name	Direction	Description
outputLevel	in	The desired 3G-SDI output level.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The connected switcher does not support 3G-SDI output.

2.3.2.15 **IBMDSwitcher::DoesSupportColorimetrySetting** method

The **DoesSupportColorimetrySetting** method determines if setting the video output colorimetry mode is supported by the switcher.

Syntax

```
HRESULT DoesSupportColorimetrySetting(Boolean* supported)
```

Parameters

Name	Direction	Description
supported	out	A Boolean value indicating whether the switcher supports changing colorimetry mode.

Return Values

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

2.3.2.16 **IBMDSwitcher::GetColorimetryMode** method

The `GetColorimetryMode` method returns the current video output colorimetry mode of the switcher.

Syntax

```
HRESULT GetColorimetryMode(BMDSwitcherColorimetryMode* colorimetry)
```

Parameters

Name	Direction	Description
colorimetry	out	The current colorimetry mode.

Return Values

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

2.3.2.17 **IBMDSwitcher::SetColorimetryMode** method

The `SetColorimetryMode` method sets the video output colorimetry mode of the switcher.

Syntax

```
HRESULT SetColorimetryMode(BMDSwitcherColorimetryMode colorimetry)
```

Parameters

Name	Direction	Description
colorimetry	in	The colorimetry mode to be set.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The colorimetry parameter is invalid.
E_NOTIMPL	The switcher does not support changing the colorimetry mode.
E_FAIL	Failure.

2.3.2.18 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.

2.3.2.19 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.

2.3.2.20 IBMDSwitcher::GetTimeCode method

The `GetTimeCode` method returns the timecode that was last received from the switcher.

Syntax

```
HRESULT GetTimeCode(uint8_t* hours, uint8_t* minutes,  
uint8_t* seconds, uint8_t* frames, boolean* dropFrame)
```

Parameters

Name	Direction	Description
hours	out	The hours value of the timecode.
minutes	out	The minutes value of the timecode.
seconds	out	The seconds value of the timecode.
frames	out	The frames value of the timecode.
dropFrame	out	Whether the timecode is drop frame.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No timecode has been received from the switcher.
E_POINTER	A parameter is invalid.

2.3.2.21 IBMDSwitcher::SetTimeCode method

The `SetTimeCode` method sets the timecode of the switcher.

Syntax

```
HRESULT SetTimeCode(uint8_t hours, uint8_t minutes, uint8_t seconds, uint8_t frames)
```

Parameters

Name	Direction	Description
hours	in	The hours value of the timecode.
minutes	in	The minutes value of the timecode.
seconds	in	The seconds value of the timecode.
frames	in	The frames value of the timecode.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	A parameter is not a valid value.
E_FAIL	Failure.

2.3.2.22 IBMDSwitcher::RequestTimeCode method

The `RequestTimeCode` method requests the current timecode from the switcher which will be cached when received. Use the `GetTimeCode` method to get the cached timecode.

Syntax

```
HRESULT RequestTimeCode()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.2.23 IBMDSwitcher::GetTimeCodeLocked method

The `GetTimeCodeLocked` method indicates whether the timecode can be changed with `SetTimeCode`.

Syntax

```
HRESULT GetTimeCodeLocked(boolean* timeCodeLocked)
```

Parameters

Name	Direction	Description
timeCodeLocked	out	The current timecode locked flag.

Return Values

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

2.3.2.24 IBMDSwitcher::GetTimeCodeMode method

The `GetTimeCodeMode` method returns the current timecode mode of the switcher.

Syntax

```
HRESULT GetTimeCodeMode(BMDSwitcherTimeCodeMode* timeCodeMode)
```

Parameters

Name	Direction	Description
timeCodeMode	out	The current timecode mode.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The timeCodeMode parameter is invalid.
S_FALSE	The switcher does not support timecode mode functionality.

2.3.2.25 IBMDSwitcher::SetTimeCodeMode method

The `SetTimeCodeMode` method sets the timecode mode of the switcher.

Syntax

```
HRESULT SetTimeCodeMode(BMDSwitcherTimeCodeMode timeCodeMode)
```

Parameters

Name	Direction	Description
timeCodeMode	in	The timecode mode to be set.

Return Values

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

2.3.2.26 IBMDSwitcher::GetAreOutputsConfigurable method

The `GetAreOutputsConfigurable` method indicates whether all of the switcher's outputs can be configured. Some switchers have mostly fixed outputs and only a small number of configurable outputs. Other switchers only have configurable outputs.

Syntax

```
HRESULT GetAreOutputsConfigurable(boolean* configurable)
```

Parameters

Name	Direction	Description
configurable	out	Boolean that indicates if the switcher only has configurable outputs.

Return Values

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

2.3.2.27 IBMDSwitcher::GetSuperSourceCascade method

The `GetSuperSourceCascade` method indicates whether the SuperSource cascade mode is currently enabled.

Syntax

```
HRESULT GetSuperSourceCascade(boolean* cascade)
```

Parameters

Name	Direction	Description
cascade	out	The current SuperSource cascade flag.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support SuperSource cascade.
E_POINTER	The cascade parameter is invalid.

2.3.2.28 IBMDSwitcher::SetSuperSourceCascade method

The `SetSuperSourceCascade` method is used to enable or disable the SuperSource cascade mode.

Syntax

```
HRESULT SetSuperSourceCascade(boolean cascade)
```

Parameters

Name	Direction	Description
cascade	in	The desired SuperSource cascade flag.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support SuperSource cascade.
E_FAIL	Failure.

2.3.2.29 IBMDSwitcher::SuspendStreaming method

The `SuspendStreaming` method is used to suspend the USB media streaming for a specified duration. The streaming will be resumed automatically.

Syntax

```
HRESULT SuspendStreaming(uint32_t durationMs)
```

Parameters

Name	Direction	Description
durationMs	in	The desired duration of the streaming suspension.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support <code>SuspendStreaming</code> .
E_FAIL	Failure.

2.3.2.30 IBMDSwitcher::AllowStreamingToResume method

The `AllowStreamingToResume` method is used to allow the client to resume the USB media streaming if it is suspended. If it is not suspended this method does nothing.

Syntax

```
HRESULT AllowStreamingToResume()
```

Return Values

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

2.3.2.31 IBMDSwitcher::DoesSupportFadeToBlackEnabledSetting method

The `DoesSupportFadeToBlackEnabledSetting` method determines if the switcher supports enabling and disabling Fade To Black control.

Syntax

```
HRESULT DoesSupportFadeToBlackEnabledSetting(Boolean* supported)
```

Parameters

Name	Direction	Description
supported	out	A Boolean value indicating whether the switcher supports disabling Fade To Black control.

Return Values

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

2.3.2.32 IBMDSwitcher::GetFadeToBlackEnabled method

The `GetFadeToBlackEnabled` method indicates whether Fade To Black control is currently enabled.

Syntax

```
HRESULT GetFadeToBlackEnabled(Boolean* enabled)
```

Parameters

Name	Direction	Description
supported	out	A Boolean value indicating whether Fade To Black control is enabled.

Return Values

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

2.3.2.33 IBMDSwitcher::SetFadeToBlackEnabled method

The `SetFadeToBlackEnabled` method is used to enable or disable Fade To Black control. Fade To Black can only be activated while this setting is enabled.

Syntax

```
HRESULT SetFadeToBlackEnabled(Boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	A Boolean value indicating whether Fade To Black control should be enabled.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support disabling Fade To Black control.
E_FAIL	Failure.

2.3.2.34 IBMDSwitcher::DoesSupportAutoVideoMode method

The `DoesSupportAutoVideoMode` method determines if auto video mode is supported by the switcher.

Syntax

```
HRESULT DoesSupportAutoVideoMode(Boolean* supported)
```

Parameters

Name	Direction	Description
supported	out	A Boolean value indicating whether the switcher supports auto video mode.

Return Values

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

2.3.2.35 IBMDSwitcher::GetAutoVideoMode method

The `GetAutoVideoMode` method indicates whether auto video mode is currently enabled.

Syntax

```
HRESULT GetAutoVideoMode(Boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	A Boolean value indicating whether auto video mode is enabled.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support auto video mode.
E_POINTER	The enabled parameter is invalid.

2.3.2.36 IBMDSwitcher::GetAutoVideoModeDetected method

The `GetAutoVideoModeDetected` method indicates whether an input video mode has been detected.

Syntax

```
HRESULT GetAutoVideoModeDetected(Boolean* detected)
```

Parameters

Name	Direction	Description
detected	out	A Boolean value indicating whether an input video mode has been detected.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support auto video mode.
E_POINTER	The detected parameter is invalid.

2.3.2.37 IBMDSwitcher::SetAutoVideoMode method

The `SetAutoVideoMode` method is used to enable or disable auto video mode.

Syntax

```
HRESULT SetAutoVideoMode(Boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	A Boolean value indicating whether auto video mode should be enabled.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support auto video mode.

2.3.2.38 IBMDSwitcher::DoesSupportDskTallyOverride method

The **DoesSupportDskTallyOverride** method determines if the switcher supports the detection of fully opaque downstream keys and will override tally when such a key is on air. Fully opaque downstream keys will cover all other on air video sources.

If this feature is enabled and a fully opaque downstream key fills the program output, tally will be updated as follows:

- Any source that was on program and is now covered by the opaque key will receive Preview tally instead of Program tally.
- Any source on preview will not be tallied.
- When the opaque downstream key is taken off air, normal tally will be restored.

Syntax

HRESULT DoesSupportDskTallyOverride(Boolean* supported)

Parameters

Name	Direction	Description
supported	out	A Boolean value indicating whether the switcher supports Downstream Key Tally Override.

Return Values

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

2.3.2.39 IBMDSwitcher::GetDskTallyOverrideEnabled method

The **GetDskTallyOverrideEnabled** method indicates whether the Downstream Key Tally Override feature is currently enabled.

Syntax

HRESULT GetDskTallyOverrideEnabled(Boolean* enabled)

Parameters

Name	Direction	Description
enabled	out	A Boolean value indicating whether Downstream Key Tally Override is enabled.

Return Values

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

2.3.2.40 **IBMDSwitcher::SetDskTallyOverrideEnabled** method

The `SetDskTallyOverride` method is used to enable or disable the Downstream Key Tally Override feature.

Syntax

```
HRESULT SetDskTallyOverrideEnabled(Boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	A Boolean value indicating whether Downstream Key Tally Override should be enabled.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support Downstream Key Tally Override.
E_FAIL	Failure.

2.3.2.41 **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	out	Pointer to returned interface object.

Return Values

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

2.3.2.42 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 IBMDSwitcherCallback object interface.

Return Values

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

2.3.2.43 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 IBMDSwitcherCallback object interface.

Return Values

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

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 <code>IBMDSwitcherCallback</code> object interface is installed with <code>IBMDSwitcher::AddCallback</code> and removed with <code>IBMDSwitcher::RemoveCallback</code>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

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	<code>BMDSwitcherEventType</code> that describes the type of event that has occurred.
coreVideoMode	in	Video standard for which the event was triggered. This parameter is used in <code>bmdSwitcherEventTypeDownConvertedHDVideoModeChanged</code> and <code>bmdSwitcherEventTypeMultiViewVideoModeChanged</code> event types.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.4 IBMDSwitcherInputIterator Interface

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

Related Interfaces

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

Public Member Functions

Method	Description
<code>Next</code>	Returns the next available switcher input.
<code>GetById</code>	Returns a switcher input for an input Id.

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
<code>input</code>	out	<code>IBMDSwitcherInput</code> object interface.

Return Values

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

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	BMDSwitcherInputId of input.
input	out	IBMDSwitcherInput object interface.

Return Values

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

2.3.5 IBMDSwitcherInput Interface

The `IBMDSwitcherInput` object interface represents a single video input into the switcher. This could be an input from a physical connector, such as SDI or HDMI, or an internally generated input such as color or a media player. Auxiliary outputs are also represented by `IBMDSwitcherInput` due to the way they are internally handled by the switcher.

To determine what type of input an `IBMDSwitcherInput` object is, call `GetPortType`.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputIterator	IID_IBMDSwitcherInputIterator	An <code>IBMDSwitcherInput</code> interface can be obtained with <code>IBMDSwitcherInputIterator::Next</code> .

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 <code>BMDSwitcherPortType</code> .
GetInputAvailability	Get the outputs this input can be routed to, as a <code>BMDSwitcherInputAvailability</code> 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.

Public Member Functions	
Method	Description
GetLongName	Get the long name describing the switcher input as a Unicode string limited to 20 bytes.
AreNamesDefault	Determine if the long name and short name are both currently set to the factory default values.
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 BMDSwitcherExternalPortType .
SetCurrentExternalPortType	Set the external port type for this input using a BMDSwitcherExternalPortType .
DoesSupportCameraModel	Returns a flag indicating whether this input supports camera model assignment.
SetCameraModel	Set the camera model for this input.
GetCameraModel	Get the current camera model for this input.
SetViscaDeviceId	Set the VISCA device for this input.
GetViscaDeviceId	Get the current VISCA device for this input.

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 IBMDSwitcherInputCallback object interface.

Return Values

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

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(IBMDSwitcherInputId* 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.

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.

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 bitwise-ANDed 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.

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.

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.

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.

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.

2.3.5.10 IBMDSwitcherInput::AreNamesDefault method

The `AreNamesDefault` method is used to check whether the long name and short name for this input are both set to the factory defaults.

Syntax

```
HRESULT AreNamesDefault(bool* isDefault)
```

Parameters

Name	Direction	Description
isDefault	out	Boolean value indicating whether the long name and short name are both set to the factory defaults.

Return Values

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

2.3.5.11 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.

2.3.5.12 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.

2.3.5.13 **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.

2.3.5.14 **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 bitwise-ANDed 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 <code>BMDSwitcherExternalPortType</code> .

Return Values

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

2.3.5.15 **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.

2.3.5.16 **IBMDSwitcherInput::DoesSupportCameraModel** method

The **DoesSupportCameraModel** method is used to determine whether assigning a camera model ID is supported by the input.

Syntax

```
HRESULT DoesSupportCameraModel(Boolean* doesSupportCameraModel)
```

Parameters

Name	Direction	Description
doesSupportCameraModel	out	Boolean value describing whether camera model assignment is supported by this input.

Return Values

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

2.3.5.17 **IBMDSwitcherInput::SetCameraModel** method

The **SetCameraModel** method assigns a camera model ID to the switcher input. The camera model ID can be any unsigned integer, there are no predefined values, but is typically assigned Blackmagic Camera IDs. The switcher does not use the camera model ID for any purpose, except to store and broadcast it to connected applications, who may use it to infer camera control characteristics.

Syntax

```
HRESULT SetCameraModel(uint32_t cameraId)
```

Parameters

Name	Direction	Description
cameraId	in	The ID indicating a camera model/type.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.5.18 **IBMDSwitcherInput::GetCameraModel** method

The **GetCameraModel** method returns the camera model ID assigned to the switcher input. The camera model ID can be any unsigned integer, there are no predefined values, but is typically assigned Blackmagic Camera IDs. The switcher does not use the camera model ID for any purpose, except to broadcast it to connected applications, who may use it to infer camera control characteristics.

Syntax

```
HRESULT GetCameraModel(uint32_t* cameraId)
```

Parameters

Name	Direction	Description
cameraId	out	The ID indicating a camera model/type.

Return Values

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

2.3.5.19 **IBMDSwitcherInput::SetViscaDeviceId** method

The `SetViscaDeviceId` method assigns a VISCA device to the input. Use a value of `bmdSwitcherViscaDeviceIdNone` to indicate no VISCA device is assigned.

Syntax

```
HRESULT SetViscaDeviceId(BMDSwitcherViscaDeviceId viscaDeviceId)
```

Parameters

Name	Direction	Description
viscaDeviceId	in	The <code>BMDSwitcherViscaDeviceId</code> of the VISCA device to assign.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.5.20 **IBMDSwitcherInput::GetViscaDeviceId** method

The `GetViscaDeviceId` method returns the VISCA device ID assigned to the input. A value of `bmdSwitcherViscaDeviceIdNone` indicates that no VISCA device is assigned.

Syntax

```
HRESULT GetViscaDeviceId(BMDSwitcherViscaDeviceId* viscaDeviceId)
```

Parameters

Name	Direction	Description
viscaDeviceId	out	The <code>BMDSwitcherViscaDeviceId</code> of the currently assigned VISCA device.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The <code>viscaDeviceId</code> parameter is invalid.

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
<code>IBMDSwitcherInput</code>	<code>IID_IBMDSwitcherInput</code>	An <code>IBMDSwitcherInputCallback</code> object interface may be installed with <code>IBMDSwitcherInput::AddCallback</code>

Public Member Functions	
Method	Description
<code>Notify</code>	A Switcher Input event occurred such as a property change.

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
<code>eventType</code>	in	<code>BMDSwitcherInputEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>E_FAIL</code>	Failure.
<code>S_OK</code>	Success.

2.3.7 IBMDSwitcherMixEffectBlockIterator Interface

The `IBMDSwitcherMixEffectBlockIterator` object interface is used to enumerate the available mix effect blocks.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::CreateIterator returns an <code>IBMDSwitcherMixEffectBlockIterator</code> object interface when the <code>IID_IBMDSwitcherMixEffectBlockIterator</code> IID is specified.

Public Member Functions

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

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	IBMDSwitcherMixEffectBlock 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.

2.3.8 IBMDSwitcherMixEffectBlock Interface

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

Related Interfaces

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

Public Member Functions	
Method	Description
<code>GetProgramInput</code>	Get the current program input.
<code>SetProgramInput</code>	Set the program input.
<code>GetPreviewInput</code>	Get the current preview input.
<code>SetPreviewInput</code>	Set the preview input.
<code>GetPreviewLive</code>	Get the current preview-live flag.
<code>GetPreviewTransition</code>	Get the current preview-transition flag.
<code>SetPreviewTransition</code>	Set the preview-transition flag.
<code>PerformAutoTransition</code>	Initiate an automatic transition.
<code>PerformCut</code>	Initiate a cut.
<code>GetInTransition</code>	Get the current in-transition flag.
<code>GetTransitionPosition</code>	Get the current transition position value.
<code>SetTransitionPosition</code>	Set the transition position value.
<code>GetTransitionFramesRemaining</code>	Get the number of transition frames remaining.
<code>PerformFadeToBlack</code>	Initiate a fade to black.
<code>GetFadeToBlackRate</code>	Get the current fade to black rate value.
<code>SetFadeToBlackRate</code>	Set the fade to black rate value.
<code>GetFadeToBlackFramesRemaining</code>	Get the current number of fade to black frames remaining.
<code>GetFadeToBlackFullyBlack</code>	Get the current fade-to-black-fully-black flag.
<code>SetFadeToBlackFullyBlack</code>	Set the fade-to-black-fully-black flag.
<code>GetInFadeToBlack</code>	Get the current in-fade-to-black flag.
<code>GetFadeToBlackInTransition</code>	Get the current fade-to-black-in-transition flag.
<code>GetInputAvailabilityMask</code>	Get the switcher input availability mask.
<code>CreateIterator</code>	Create an iterator.
<code>AddCallback</code>	Add a callback to receive mix effect block events.
<code>RemoveCallback</code>	Remove a callback.

2.3.8.1 **IBMDSwitcherMixEffectBlock::GetProgramInput** method

The `GetProgramInput` method returns the current program input to the mix effect block.

Syntax

```
HRESULT GetProgramInput(BMDSwitcherInputId* value)
```

Parameters

Name	Direction	Description
value	out	The program input currently applied to the mix effect block.

Return Values

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

2.3.8.2 **IBMDSwitcherMixEffectBlock::SetProgramInput** method

The `SetProgramInput` method sets the program input to apply to the mix effect block.

Syntax

```
HRESULT SetProgramInput(BMDSwitcherInputId value)
```

Parameters

Name	Direction	Description
value	in	The program input to apply to the mix effect block.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure
E_INVALIDARG	The value is not a valid identifier.

2.3.8.3 IBMDSwitcherMixEffectBlock::GetPreviewInput method

The `GetPreviewInput` method returns the current preview input to the mix effect block.

Syntax

```
HRESULT GetPreviewInput(BMDSwitcherInputId* value)
```

Parameters

Name	Direction	Description
value	out	The preview input currently applied to the mix effect block.

Return Values

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

2.3.8.4 IBMDSwitcherMixEffectBlock::SetPreviewInput method

The `SetPreviewInput` method sets the preview input to apply to the mix effect block.

Syntax

```
HRESULT SetPreviewInput(BMDSwitcherInputId value)
```

Parameters

Name	Direction	Description
value	in	The preview input to apply to the mix effect block

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The value is not a valid identifier.

2.3.8.5 IBMDSwitcherMixEffectBlock::GetPreviewLive method

The `GetPreviewLive` method indicates whether the preview is live.

Syntax

```
HRESULT GetPreviewLive(BOOLEAN* value)
```

Parameters

Name	Direction	Description
value	out	The preview live flag.

Return Values

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

2.3.8.6 IBMDSwitcherMixEffectBlock::GetPreviewTransition method

The `GetPreviewTransition` method indicates whether the preview transition mode is currently enabled on the mix effect block.

Syntax

```
HRESULT GetPreviewTransition(BOOLEAN* value)
```

Parameters

Name	Direction	Description
value	out	The current preview transition flag.

Return Values

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

2.3.8.7 IBMDSwitcherMixEffectBlock::SetPreviewTransition method

The `SetPreviewTransition` method is used to enable or disable the preview transition mode.

Syntax

```
HRESULT SetPreviewTransition(BOOLEAN value)
```

Parameters

Name	Direction	Description
value	in	The desired preview transition flag

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.8.8 **IBMDSwitcherMixEffectBlock::PerformAutoTransition** method

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

When the transition begins and ends the `bmdSwitcherMixEffectBlockEventTypeInTransitionChanged` callback will be fired with the in transition flag set to true and then false on completion. Throughout the transition the `bmdSwitcherMixEffectBlockEventTypeTransitionPositionChanged` and `bmdSwitcherMixEffectBlockEventTypeTransitionFramesRemainingChanged` callbacks will be fired with property values corresponding to the progress through the transition.

Syntax

```
HRESULT PerformAutoTransition()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.8.9 **IBMDSwitcherMixEffectBlock::PerformCut** method

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

Syntax

```
HRESULT PerformCut()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.8.10 **IBMDSwitcherMixEffectBlock::GetInTransition** method

The `GetInTransition` method indicates whether a transition is occurring

Syntax

```
HRESULT GetInTransition(boolean* value)
```

Parameters

Name	Direction	Description
value	out	The current in transition flag.

Return Values

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

2.3.8.11 **IBMDSwitcherMixEffectBlock::GetTransitionPosition** method

The `GetTransitionPosition` method returns the current transition position value.

Syntax

```
HRESULT GetTransitionPosition(double* value)
```

Parameters

Name	Direction	Description
value	out	The current transition position value.

Return Values

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

2.3.8.12 **IBMDSwitcherMixEffectBlock::SetTransitionPosition** method

The `SetTransitionPosition` method sets the transition position value.

Syntax

```
HRESULT SetTransitionPosition(double value)
```

Parameters

Name	Direction	Description
value	in	The desired transition position value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Fail.

2.3.8.13 **IBMDSwitcherMixEffectBlock::GetTransitionFramesRemaining** method

The `GetTransitionFramesRemaining` method returns the number of transition frames remaining for the transition.

Syntax

```
HRESULT GetTransitionFramesRemaining(uint32_t* value)
```

Parameters

Name	Direction	Description
value	out	The number of transition frames remaining.

Return Values

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

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 and ends the **bmdSwitcherMixEffectBlockEventTypeInFadeToBlackChanged** callback will be fired with the in fade to black flag set to true and then false on completion. Throughout the transition the **bmdSwitcherMixEffectBlockEventTypeFadeToBlackFramesRemainingChanged** callback will be fired with values corresponding to the progress through the fade to black.

Syntax

```
HRESULT PerformFadeToBlack()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.8.15 **IBMDSwitcherMixEffectBlock::GetFadeToBlackRate** method

The **GetFadeToBlackRate** method returns the current fade to black rate in frames.

Syntax

```
HRESULT GetFadeToBlackRate(uint32_t* value)
```

Parameters

Name	Direction	Description
value	out	The current fade to black rate.

Return Values

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

2.3.8.16 **IBMDSwitcherMixEffectBlock::SetFadeToBlackRate** method

The `SetFadeToBlackRate` method returns the number of frames remaining for the fade to black.

Syntax

```
HRESULT SetFadeToBlackRate(uint32_t value)
```

Parameters

Name	Direction	Description
value	in	The desired fade to black rate.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The value is not a valid number of frames.

2.3.8.17 **IBMDSwitcherMixEffectBlock::GetFadeToBlackFramesRemaining** method

The `GetFadeToBlackFramesRemaining` method returns the number of frames remaining for the fade to black.

Syntax

```
HRESULT GetFadeToBlackFramesRemaining(uint32_t* value)
```

Parameters

Name	Direction	Description
value	out	The number of fade to black frames remaining.

Return Values

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

2.3.8.18 **IBMDSwitcherMixEffectBlock::GetFadeToBlackFullyBlack** method

The `GetFadeToBlackFullyBlack` method indicates whether the current frame is completely black.

Syntax

```
HRESULT GetFadeToBlackFullyBlack(boolean* value)
```

Parameters

Name	Direction	Description
value	out	The current fade to black fully black flag.

Return Values

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

2.3.8.19 **IBMDSwitcherMixEffectBlock::SetFadeToBlackFullyBlack** method

The `SetFadeToBlackFullyBlack` method sets the fade to black fully black flag.

Syntax

```
HRESULT SetFadeToBlackFullyBlack(boolean value)
```

Parameters

Name	Direction	Description
value	in	The desired fade to black fully black flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.8.20 **IBMDSwitcherMixEffectBlock::GetInFadeToBlack** method

The `GetInFadeToBlack` method indicates whether fade to black is transitioning or the frame is completely black.

Syntax

```
HRESULT GetInFadeToBlack(boolean* value)
```

Parameters

Name	Direction	Description
value	out	The current in fade to black flag.

Return Values

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

2.3.8.21 **IBMDSwitcherMixEffectBlock::GetFadeToBlackInTransition** method

The `GetFadeToBlackInTransition` method indicates whether fade to black is transitioning.

Syntax

```
HRESULT GetFadeToBlackInTransition(boolean* value)
```

Parameters

Name	Direction	Description
value	out	The current fade to black in transition flag.

Return Values

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

2.3.8.22 **IBMDSwitcherMixEffectBlock::GetInputAvailabilityMask** method

The `GetInputAvailabilityMask` method returns the corresponding `BMDSwitcherInputAvailability` bit mask value for this mix effect block. The input availability property 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 by this mix effect block.

Syntax

```
HRESULT GetInputAvailabilityMask(BMDSwitcherInputAvailability* value)
```

Parameters

Name	Direction	Description
value	out	The availability of the input.

Return Values

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

2.3.8.23 **IBMDSwitcherMixEffectBlock::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	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.
E_NOINTERFACE	Interface was not found.

2.3.8.24 **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 IBMDSwitcherMixEffectBlockCallback object interface.

Return Values

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

2.3.8.25 **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.

2.3.9 IBMDSwitcherMixEffectBlockCallback Interface

The `IBMDSwitcherMixEffectBlockCallback` object interface is a callback class containing methods that are called when an event occurs on an `IBMDSwitcherMixEffectBlock` object.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherMixEffectBlock</code>	<code>IID_IBMDSwitcherMixEffectBlock</code>	An <code>IBMDSwitcherMixEffectBlockCallback</code> object interface is installed with <code>IBMDSwitcherMixEffectBlock::AddCallback</code> and removed with <code>IBMDSwitcherMixEffectBlock::RemoveCallback</code> .

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

2.3.9.1 IBMDSwitcherMixEffectBlockCallback::Notify method

The `Notify` method is called when `IBMDSwitcherMixEffectBlock` 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(BMDSwitcherMixEffectBlockEventType eventType)
```

Parameters

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

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

2.3.10 IBMDSwitcherInputColor Interface

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

Related Interfaces

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

Public Member Functions

Method	Description
<code>GetHue</code>	Get the current hue value.
<code>SetHue</code>	Set the hue value.
<code>GetSaturation</code>	Get the current saturation value.
<code>SetSaturation</code>	Set the saturation value.
<code>GetLuma</code>	Get the current luma value.
<code>SetLuma</code>	Set the luma value.
<code>AddCallback</code>	Add a color input callback.
<code>RemoveCallback</code>	Remove a color input callback.

2.3.10.1 IBMDSwitcherInputColor::GetHue method

The `GetHue` method gets the current hue value.

Syntax

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

Parameters

Name	Direction	Description
<code>hue</code>	in	The current hue value.

Return Values

Value	Description
<code>E_POINTER</code>	The hue parameter is invalid.
<code>S_OK</code>	Success.

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 saturation value.

Return Values

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

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 luma value.

Syntax

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

Parameters

Name	Direction	Description
luma	out	The current luma value.

Return Values

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

2.3.10.6 IBMDSwitcherInputColor::SetLuma method

The SetLuma method sets the luma value.

Syntax

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

Parameters

Name	Direction	Description
luma	in	The desired luma value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

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 <code>IBMDSwitcherInputColorCallback</code> object interface.

Return Values

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

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 <code>IBMDSwitcherInputColorCallback</code> object interface.

Return Values

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

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
<code>IBMDSwitcherInputColor</code>	<code>IID_IBMDSwitcherInputColor</code>	An <code>IBMDSwitcherInputColorCallback</code> object interface is installed with <code>IBMDSwitcherInputColor::AddCallback</code> and removed with <code>IBMDSwitcherInputColor::RemoveCallback</code>

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

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
<code>eventType</code>	in	<code>BMDSwitcherInputColorEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

2.3.12 IBMDSwitcherInputAux Interface

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

Related Interfaces

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

Public Member Functions

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

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
<code>inputId</code>	out	The <code>BMDSwitcherInputId</code> of the selected input source.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	Invalid <code>inputId</code> parameter.

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 <code>BMDSwitcherInputId</code> of the desired input source.

Return Values

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

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 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 by this auxiliary port.

Syntax

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

Parameters

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

Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

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 IBMDSwitcherInputAuxCallback object interface.

Return Values

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

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 IBMDSwitcherInputAuxCallback object interface.

Return Values

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

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
<code>IBMDSwitcherInputAux</code>	<code>IID_IBMDSwitcherInputAux</code>	An <code>IBMDSwitcherInputAuxCallback</code> object interface is installed with <code>IBMDSwitcherInputAux::AddCallback</code> and removed with <code>IBMDSwitcherInputAux::RemoveCallback</code>

Public Member Functions	
Method	Description
<code>Notify</code>	Called when an event occurs.

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
<code>eventType</code>	in	<code>BMDSwitcherInputAuxEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

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
<code>IBMDSwitcher</code>	<code>IID_IBMDSwitcher</code>	<code>IBMDSwitcher::CreateIterator</code> can return an <code>IBMDSwitcherMultiViewIterator</code> object interface.

Public Member Functions

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

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
<code>multiView</code>	out	<code>IBMDSwitcherMultiView</code> object interface.

Return Values

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

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
<code>IBMDSwitcherMultiViewIteator</code>	<code>IID_IBMDSwitcherMultiViewIteator</code>	An <code>IBMDSwitcherMultiView</code> object will be returned after a successful call to <code>IBMDSwitcherMultiViewIteator::Next</code> method.

Public Member Functions	
Method	Description
<code>CanChangeLayout</code>	Determine if the Multiview supports layout changing.
<code>GetLayout</code>	Get the current layout format.
<code>SetLayout</code>	Set layout format.
<code>SupportsQuadrantLayout</code>	Determine if this MultiView supports quadrant layout.
<code>GetWindowInput</code>	Get current input routing of a window.
<code>SetWindowInput</code>	Set input routing of a window.
<code>GetWindowCount</code>	Get number of windows available.
<code>GetInputAvailabilityMask</code>	Get the corresponding <code>BMDSwitcherInputAvailability</code> bit value for MultiView.
<code>CanRouteInputs</code>	Determine if this MultiView supports custom input-to-window routing.
<code>SupportsVuMeters</code>	Determine if this switcher supports the display of VU meters on the MultiView.
<code>CurrentInputSupportsVuMeter</code>	Check if the current input of a specified MultiView window supports VU meters.
<code>GetVuMeterEnabled</code>	Check if the VU meter is currently visible on a specified window.
<code>SetVuMeterEnabled</code>	Hide or show the VU meter on a specified window.
<code>CanAdjustVuMeterOpacity</code>	Determine if this MultiView supports changing the VU meter opacity.
<code>GetVuMeterOpacity</code>	Get the current MultiView VU meter opacity.
<code>SetVuMeterOpacity</code>	Set the MultiView VU meter opacity.
<code>CanToggleSafeAreaEnabled</code>	Determine if this MultiView supports toggling the safe area overlay on and off.
<code>CurrentInputSupportsSafeArea</code>	Determine if a MultiView window supports displaying the safe area overlay.
<code>GetSafeAreaEnabled</code>	Check if the safe area overlay is currently visible.
<code>SetSafeAreaEnabled</code>	Hide or show the safe area overlay.
<code>GetSupportedSafeAreaTypes</code>	Get the supported safe area overlay types.
<code>GetSafeAreaType</code>	Get the current safe area overlay type.
<code>SetSafeAreaType</code>	Set the safe area overlay type.
<code>SupportsProgramPreviewSwap</code>	Determine if this MultiView supports swapping the positions of the program and preview windows.

Public Member Functions	
Method	Description
GetProgramPreviewSwapped	Check if the program and preview window positions are currently swapped.
SetProgramPreviewSwapped	Set the program and preview window positions to standard or swapped.
CanChangeOverlayProperties	Determine if the MultiView supports changing the overlay label and border properties.
CurrentInputSupportsLabelOverlay	Check if the label overlay is supported on the specified window.
GetLabelVisible	Check if the label is currently visible on a specified window.
SetLabelVisible	Hide or show the label on a specified window.
GetBorderVisible	Check if the border is currently visible on a specified window.
SetBorderVisible	Hide or show the border on a specified window.
GetBorderColor	Get the current overlay border color.
SetBorderColor	Set the overlay border color.
AddCallback	Add a MultiView callback.
RemoveCallback	Remove a MultiView callback.

2.3.15.1 IBMDSwitcherMultiView::CanChangeLayout method

The `CanChangeLayout` method is used to determine if the switcher supports changing the MultiView window layout.

Most switchers are capable of changing the layout of the windows displayed in the MultiView.

Syntax

```
HRESULT CanChangeLayout(BOOLEAN* canChangeLayout)
```

Parameters

Name	Direction	Description
canChangeLayout	out	A Boolean value indicating whether the switcher supports changing the MultiView layout.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The <code>canChangeLayout</code> parameter is invalid.

2.3.15.2 **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 <code>BMDSwitcherMultiViewLayout</code> .

Return Values

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

2.3.15.3 **IBMDSwitcherMultiView::SetLayout** method

The `SetLayout` method sets the layout format.

If the switcher supports quadrant layout, then `bmdSwitcherMultiViewLayoutTopLeftSmall`, `bmdSwitcherMultiViewLayoutTopRightSmall`, `bmdSwitcherMultiViewLayoutBottomLeftSmall`, and `bmdSwitcherMultiViewLayoutBottomRightSmall` are bitmask fields that can be bitwise-ORed in any combination to describe which quadrants should show four small windows. If the bit for a quadrant is not set, the quadrant will display a large window.

If a switcher does not support quadrant layout, then only `bmdSwitcherMultiViewLayoutProgramTop`, `bmdSwitcherMultiViewLayoutProgramBottom`, `bmdSwitcherMultiViewLayoutProgramLeft`, and `bmdSwitcherMultiViewLayoutProgramRight` are valid.

Syntax

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

Parameters

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

Return Values

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

2.3.15.4 IBMDSwitcherMultiView::SupportsQuadrantLayout method

The `SupportsQuadrantLayout` method is used to determine if the switcher supports quadrant layout.

Some switchers are capable of configuring each quadrant of the `MultiView` as either one large window or four small windows. Switchers that are not capable of independent quadrant configuration are still capable of displaying the classic ten window configuration which is made up of two large windows either at the top, bottom, left or right side of the display and eight small windows in the remaining area of the display.

Syntax

```
HRESULT SupportsQuadrantLayout(boolean* supportsQuadrantLayout)
```

Parameters

Name	Direction	Description
supportsQuadrantLayout	out	Boolean value indicating whether quadrant layout is supported by the switcher.

Return Values

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

2.3.15.5 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 <code>BMDSwitcherInputId</code> .

Return Values

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

2.3.15.6 **IBMDSwitcherMultiView::SetWindowInput** method

The **SetWindowInput** method routes an input source to the specified window. Note that for switchers that do not support quadrant layout, the inputs for windows 0 and 1 are reserved for the Preview and Program outputs, and so cannot be set using this method. In this case calling this method with a window index of 0 or 1 will do nothing and will return **S_FALSE**. For switchers that do support quadrant layout, inputs for windows 0 and 1 may be assigned as normal.

Syntax

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

Parameters

Name	Direction	Description
window	in	Zero-based window index.
input	in	BMDSwitcherInputId input source.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The input for the specified window cannot be set using this method.
E_FAIL	Failure.
E_INVALIDARG	The window and/or inputId parameter is invalid.

2.3.15.7 **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.

2.3.15.8 **IBMDSwitcherMultiView::GetInputAvailabilityMask** method

The **GetInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for this MultiView. The input availability property 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 viewing in a window.

Syntax

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

Parameters

Name	Direction	Description
mask	out	BMDSwitcherInputAvailability bit mask.

Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

2.3.15.9 **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.

2.3.15.10 IBMDSwitcherMultiView::SupportsVuMeters method

The `SupportsVuMeters` method is used to determine if the switcher supports the display of VU meters on the MultiView.

Syntax

```
HRESULT SupportsVuMeters(boolean* supportsVuMeters)
```

Parameters

Name	Direction	Description
supportsVuMeters	out	Boolean value indicating whether VU meters are supported by the switcher.

Return Values

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

2.3.15.11 IBMDSwitcherMultiView::CurrentInputSupportsVuMeter method

The `CurrentInputSupportsVuMeter` method is used to determine if a MultiView window is currently set to an input that supports the display of a VU meter. Some inputs, such as Color Bars and Color Generators do not support the display of a VU meter.

Syntax

```
HRESULT CurrentInputSupportsVuMeter(int32_t window, boolean* supportsVuMeter)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
supportsVuMeter	out	Boolean value indicating whether display of a VU meter is supported by the input that is currently selected on the specified window.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support VU meters.
E_POINTER	The supportsVuMeter parameter is not a valid pointer.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.12 IBMDSwitcherMultiView::GetVuMeterEnabled method

The `GetVuMeterEnabled` method is used to determine if the the VU meter is currently visible on the specified MultiView window.

Syntax

```
HRESULT GetVuMeterEnabled(uint32_t window, boolean* enabled)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
enabled	out	Boolean value indicating whether the VU meter is currently visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support VU meters.
E_POINTER	The enabled parameter is not a valid pointer.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.13 IBMDSwitcherMultiView::SetVuMeterEnabled method

The `SetVuMeterEnabled` method is used to hide or show VU meters on the specified MultiView window.

Syntax

```
HRESULT SetVuMeterEnabled(uint32_t window, boolean enabled)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
enabled	in	Boolean value indicating whether VU meters should be made visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support VU meters.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.14 **IBMDSwitcherMultiView::CanAdjustVuMeterOpacity** method

The `CanAdjustVuMeterOpacity` method is used to determine if the switcher supports changing the MultiView window VU meter opacity.

Syntax

```
HRESULT CanAdjustVuMeterOpacity(Boolean* canAdjustVuMeterOpacity)
```

Parameters

Name	Direction	Description
canAdjustVuMeterOpacity	out	Boolean value indicating whether the switcher supports changing the VU meter opacity.

Return Values

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

2.3.15.15 **IBMDSwitcherMultiView::GetVuMeterOpacity** method

The `GetVuMeterOpacity` method returns the opacity of the VU meters displayed on the MultiView as a value between zero and one. A value of 0.0 is fully transparent, and a value of 1.0 is fully opaque.

Syntax

```
HRESULT GetVuMeterOpacity(double* opacity)
```

Parameters

Name	Direction	Description
opacity	out	The opacity of the VU meters from 0.0 (fully transparent) to 1.0 (fully opaque).

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support VU meters.
E_POINTER	The opacity parameter is not a valid pointer.

2.3.15.16 IBMDSwitcherMultiView::SetVuMeterOpacity method

The `SetVuMeterOpacity` method is used to set the opacity of the VU meters displayed on the MultiView.

Syntax

```
HRESULT SetVuMeterOpacity(double opacity)
```

Parameters

Name	Direction	Description
opacity	in	The opacity of the VU meters from 0.0 (fully transparent) to 1.0 (fully opaque).

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support VU meters.

2.3.15.17 IBMDSwitcherMultiView::CanToggleSafeAreaEnabled method

The `CanToggleSafeAreaEnabled` method is used to determine whether the switcher supports toggling the safe area overlay on the MultiView preview window on and off.

Syntax

```
HRESULT CanToggleSafeAreaEnabled(boolean* canToggleSafeAreaEnabled)
```

Parameters

Name	Direction	Description
canToggleSafeAreaEnabled	out	A Boolean value indicating whether the switcher supports toggling the safe area overlay on and off.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The <code>canToggleSafeAreaEnabled</code> parameter is not a valid pointer.

2.3.15.18 **IBMDSwitcherMultiView::CurrentInputSupportsSafeArea** method

The **CurrentInputSupportsSafeArea** method is used to determine if a MultiView window supports displaying the safe area overlay. Only large windows whose input is set to a preview output will support displaying the safe area overlay.

Syntax

```
HRESULT CurrentInputSupportsSafeArea(uint32_t window, boolean* supportsSafeArea)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
supportsSafeArea	out	Boolean value indicating whether display of a the safe area is supported by the input that is currently selected on the specified window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The supportsSafeArea parameter is invalid.
E_INVALIDARG	The window parameter is not a valid window index.
E_FAIL	Failure. This can happen if the switcher does not support toggling the safe area overlay on and off.

2.3.15.19 **IBMDSwitcherMultiView::GetSafeAreaEnabled** method

The **GetSafeAreaEnabled** method is used to determine whether the safe area overlay is currently visible on the MultiView window.

Syntax

```
HRESULT GetSafeAreaEnabled(uint32_t window, boolean* enabled)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
enabled	out	A Boolean value indicating whether the safe area overlay is currently visible on the MultiView window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The enabled parameter is not a valid pointer.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.20 **IBMDSwitcherMultiView::SetSafeAreaEnabled** method

The **SetSafeAreaEnabled** method is used to hide or show the safe area overlay on the MultiView window.

Syntax

```
HRESULT SetSafeAreaEnabled(uint32_t window, boolean* enabled)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
enabled	in	A Boolean value indicating whether the safe area overlay should be made visible on the MultiView window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The enabled parameter is invalid.
E_INVALIDARG	The window parameter is not a valid window index.
E_FAIL	Failure. This can happen if the switcher does not support toggling the safe area overlay on and off.

2.3.15.21 **IBMDSwitcherMultiView::GetSupportedSafeAreaTypes** method

The **GetSupportedSafeAreaTypes** method is used to determine which safe area overlay types are supported by the Switcher.

Syntax

```
HRESULT GetSupportedSafeAreaTypes(uint32_t* supportedSafeAreaTypeFlags)
```

Parameters

Name	Direction	Description
supportedSafeAreaTypeFlags	out	A BMDSwitcherMultiViewSafeAreaType bit mask of the supported safe area types.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The supportedSafeAreaTypeFlags parameter is invalid.
E_NOTIMPL	The switcher does not support changing the safe area overlay type.

2.3.15.22 **IBMDSwitcherMultiView::GetSafeAreaType** method

The **GetSafeAreaType** method is used to determine the type of safe area overlay currently displayed on the MultiView window.

Syntax

```
HRESULT GetSafeAreaType(uint32_t window,  
                        BMDSwitcherMultiViewSafeAreaType* type)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
type	out	The current safe area overlay type as a BMDSwitcherMultiViewSafeAreaType bit mask.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The type parameter is invalid.
E_INVALIDARG	The window parameter is not a valid window index.
E_NOTIMPL	The switcher does not support changing the safe area overlay type.

2.3.15.23 **IBMDSwitcherMultiView::SetSafeAreaType** method

The **SetSafeAreaType** method is used to set the type of safe area overlay displayed on the MultiView window.

Syntax

```
HRESULT SetSafeAreaType(uint32_t window,  
                        BMDSwitcherMultiViewSafeAreaType type)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
type	in	The desired safe area overlay type as a BMDSwitcherMultiViewSafeAreaType bit mask.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The window parameter is not a valid window index.
E_NOTIMPL	The switcher does not support changing the safe area overlay type.
E_FAIL	Failure.

2.3.15.24 IBMDSwitcherMultiView::SupportsProgramPreviewSwap method

The `SupportsProgramPreviewSwap` method is used to determine if the switcher supports swapping the positions of the program and preview windows on the MultiView. Standard positioning places the preview window to the left of or above the program window. Swapping places the preview window to the right of or below the program window.

Syntax

```
HRESULT SupportsProgramPreviewSwap(boolean* supportsProgramPreviewSwap)
```

Parameters

Name	Direction	Description
supportsProgramPreviewSwap	out	A Boolean value indicating whether the switcher supports swapping the program and preview window positions on the MultiView.

Return Values

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

2.3.15.25 IBMDSwitcherMultiView::GetProgramPreviewSwapped method

The `GetProgramPreviewSwapped` method is used to determine if the MultiView program and preview window positions are currently swapped.

Syntax

```
HRESULT GetProgramPreviewSwapped(boolean* swapped)
```

Parameters

Name	Direction	Description
swapped	out	A Boolean value indicating whether the program and preview window positions are currently swapped.

Return Values

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

2.3.15.26 **IBMDSwitcherMultiView::SetProgramPreviewSwapped** method

The **SetProgramPreviewSwapped** method is used to specify whether the MultiView program and preview window positions should be swapped.

Syntax

```
HRESULT SetProgramPreviewSwapped(boolean swapped)
```

Parameters

Name	Direction	Description
swapped	in	A Boolean value indicating whether the program and preview window positions should be swapped.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the switcher does not support swapping the positions of the program and preview windows.

2.3.15.27 **IBMDSwitcherMultiView::CanChangeOverlayProperties** method

The **CanChangeOverlayProperties** method is used to determine if the switcher supports changing the label and border visibility and border color on the MultiView.

Syntax

```
HRESULT CanChangeOverlayProperties(Boolean* canChangeOverlay)
```

Parameters

Name	Direction	Description
canChangeOverlay	out	Boolean value indicating whether it is possible to change the label and border visibility and border colour.

Return Values

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

2.3.15.28 IBMDSwitcherMultiView::CurrentInputSupportsLabelOverlay method

The `CurrentInputSupportsLabelOverlay` method is used to determine if the label overlay is supported on the specified MultiView window.

Syntax

```
HRESULT CurrentInputSupportsLabelOverlay(uint32_t window,
                                         Boolean* supportsLabelOverlay)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
supportsLabelOverlay	out	Boolean value indicating whether the label overlay is supported on the specified window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The supportsLabelOverlay parameter is invalid.
E_NOTIMPL	The switcher does not support changing the label visibility.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.29 IBMDSwitcherMultiView::GetLabelVisible method

The `GetLabelVisible` method is used to determine if the label is currently visible on the specified MultiView window.

Syntax

```
HRESULT GetLabelVisible(uint32_t window, Boolean* visible)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
visible	out	Boolean value indicating whether the label is currently visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The visible parameter is invalid.
E_NOTIMPL	The switcher does not support changing the label visibility.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.30 **IBMDSwitcherMultiView::SetLabelVisible** method

The **SetLabelVisible** method is used to hide or show the label on the specified MultiView window.

Syntax

```
HRESULT SetLabelVisible(uint32_t window, Boolean visible)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
visible	in	Boolean value indicating whether the label should be made visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support changing the label visibility.
E_INVALIDARG	The window parameter is not a valid window index.
E_FAIL	Failure.

2.3.15.31 **IBMDSwitcherMultiView::GetBorderVisible** method

The **GetBorderVisible** method is used to determine if the border is currently visible on the specified MultiView window.

Syntax

```
HRESULT GetBorderVisible(uint32_t window, Boolean* visible)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
visible	out	Boolean value indicating whether the border is currently visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The visible parameter is invalid.
E_NOTIMPL	The switcher does not support changing the border visibility.
E_INVALIDARG	The window parameter is not a valid window index.

2.3.15.32 IBMDSwitcherMultiView::SetBorderVisible method

The `SetBorderVisible` method is used to hide or show the border on the specified MultiView window.

Syntax

```
HRESULT SetBorderVisible(uint32_t window, Boolean visible)
```

Parameters

Name	Direction	Description
window	in	Zero-based window index.
visible	in	Boolean value indicating whether the border should be made visible on the specified window.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support changing the border visibility.
E_INVALIDARG	The window parameter is not a valid window index.
E_FAIL	Failure.

2.3.15.33 IBMDSwitcherMultiView::GetBorderColor method

The `GetBorderColor` method is used to determine the default border color of the MultiView.

Syntax

```
HRESULT GetBorderColor(double* red, double* green, double* blue,  
double* alpha)
```

Parameters

Name	Direction	Description
red	out	The current red component value of the color, ranging from 0.0 to 1.0.
green	out	The current green component value of the color, ranging from 0.0 to 1.0.
blue	out	The current blue component value of the color, ranging from 0.0 to 1.0.
alpha	out	The current transparent component value of the color, ranging from 0.0 to 1.0.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The red, green, blue, or alpha parameter is invalid.
E_NOTIMPL	The switcher does not support changing the border color.

2.3.15.34 IBMDSwitcherMultiView::SetBorderColor method

The `SetBorderColor` method is used to set the default border color of the MultiView. This does not effect the border colors for tally.

Syntax

```
HRESULT SetBorderColor(double red, double green, double blue, double alpha)
```

Parameters

Name	Direction	Description
red	in	The desired red component value of the color, ranging from 0.0 to 1.0.
green	in	The desired green component value of the color, ranging from 0.0 to 1.0.
blue	in	The desired blue component value of the color, ranging from 0.0 to 1.0.
alpha	in	The desired transparent component value of the color, ranging from 0.0 to 1.0.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support changing the border color.
E_FAIL	Failure.

2.3.15.35 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 <code>IBMDSwitcherMultiViewCallback</code> object interface.

Return Values

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

2.3.15.36 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 IBMDSwitcherMultiViewCallback object interface.

Return Values

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

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 IBMDSwitcherMultiViewCallback object interface is installed with IBMDSwitcherMultiView::AddCallback and removed with IBMDSwitcherMultiView::RemoveCallback

Public Member Functions

Method	Description
Notify	Called when an event occurs.

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	BMDSwitcherMultiViewEventType that describes the type of event that has occurred.
window	in	This parameter is only valid when eventType is bmdSwitcherMultiViewEventTypeWindowChanged , it specifies the window index that was changed.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

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	IBMDSwitcher::CreateIterator returns an IBMDSwitcherSerialPortIterator interface when the IID_IBMDSwitcherSerialPortIterator IID is specified.

Public Member Functions	
Method	Description
Next	Returns the next available serial port.

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	IBMDSwitcherSerialPort object interface.

Return Values

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

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	IBMDSwitcherSerialPortIterator::Next returns IBMDSwitcherSerialPort 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 BMDSwitcherSerialPortFunction .
GetFunction	Returns the function of the serial port as a BMDSwitcherSerialPortFunction .
DoesSupportFunction	Check if a given BMDSwitcherSerialPortFunction is supported by the switcher.
SetBaudRate	Set the baud rate of the serial port using a BMDSwitcherSerialPortBaudRate .
GetBaudRate	Returns the baud rate of the serial port as a BMDSwitcherSerialPortBaudRate .
AddCallback	Add a serial port callback.
RemoveCallback	Remove a serial port callback.

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.

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 <code>BMDSwitcherSerialPortFunction</code> 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.

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 BMDSwitcherSerialPortFunction .

2.3.18.4 **IBMDSwitcherSerialPort::SetBaudRate** method

The **SetBaudRate** method configures the baud rate of the serial port for the current function. Some serial port functions, such as GVG, have a fixed baud rate and ignore this setting.

Syntax

```
HRESULT SetBaudRate(BMDSwitcherSerialPortBaudRate baudRate)
```

Parameters

Name	Direction	Description
baudRate	in	The desired baud rate to set for the current function.

Return Values

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

2.3.18.5 **IBMDSwitcherSerialPort::GetBaudRate** method

The **GetBaudRate** method returns the currently configured baud rate of the serial port.

Syntax

```
HRESULT GetBaudRate(BMDSwitcherSerialPortBaudRate* baudRate)
```

Parameters

Name	Direction	Description
baudRate	out	The current baud rate for the current function.

Return Values

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

2.3.18.6 **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 IBMDSwitcherSerialPortCallback interface.

Return Values

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

2.3.18.7 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.

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 IBMDSwitcherSerialPortCallback interface may be installed with IBMDSwitcherSerialPort::AddCallback

Public Member Functions

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

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(IBMDSwitcherSerialPortEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherSerialPortEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_FAIL	Failure.
S_OK	Success.

2.3.20 IBMDSwitcherMixMinusOutputIterator Interface

The **IBMDSwitcherMixMinusOutputIterator** is used to enumerate the available mix minus outputs for the switcher.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::CreateIterator returns an IBMDSwitcherMixMinusOutputIterator interface when the IID_IBMDSwitcherMixMinusOutputIterator IID is specified.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherMixMinusOutput object interface.

2.3.20.1 IBMDSwitcherMixMinusOutputIterator::Next method

The `Next` method returns the next available `IBMDSwitcherMixMinusOutput` object interface.

The `IBMDSwitcherMixMinusOutput` object interface must be released by the caller when no longer required.

Syntax

```
HRESULT Next(IBMDSwitcherMixMinusOutput** mixMinusOutput)
```

Parameters

Name	Direction	Description
mixMinusOutput	out	IBMDSwitcherMixMinusOutput object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No (more) mix minus outputs are available.
E_POINTER	The mixMinusOutput parameter is not a valid pointer.

2.3.21 IBMDSwitcherMixMinusOutput Interface

The `IBMDSwitcherMixMinusOutput` object interface is used for managing a mix minus output port.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixMinusOutputIterator	IID_IBMDSwitcherMixMinusOutputIterator	An <code>IBMDSwitcherMixMinusOutput</code> object will be returned after a successful call to the <code>IBMDSwitcherMixMinusOutputIterator::Next</code> method.

Public Member Functions

Method	Description
GetAvailableAudioModes	Get the available mix minus output audio modes as a bit mask of <code>BMDSwitcherMixMinusOutputAudioMode</code> .
GetAudioMode	Get the current audio mode of the output as a <code>BMDSwitcherMixMinusOutputAudioMode</code> .
SetAudioMode	Set the audio mode of the output using a <code>BMDSwitcherMixMinusOutputAudioMode</code> .
HasMinusAudioInputId	Get the current has minus audio input ID flag.
GetMinusAudioInputId	Get the <code>BMDSwitcherAudioInputId</code> of the audio input that is subtracted when in mix minus audio mode.
AddCallback	Add a mix minus output callback.
RemoveCallback	Remove a mix minus output callback.

2.3.21.1 **IBMDSwitcherMixMinusOutput::GetAvailableAudioModes** method

The `GetAvailableAudioModes` method gets the available mix minus output audio modes for this switcher, given as a bit mask of `BMDSwitcherMixMinusOutputAudioMode`. This bit mask can be bitwise-ANDed with any value of `BMDSwitcherMixMinusOutputAudioMode` (e.g. `bmdSwitcherMixMinusOutputAudioModeProgramOut`) to determine if that mix minus output audio mode is available.

Syntax

```
HRESULT GetAvailableAudioModes(BMDSwitcherMixMinusOutputAudioMode* audioModes)
```

Parameters

Name	Direction	Description
audioModes	out	The available mix minus output audio modes as a bit mask of <code>BMDSwitcherMixMinusOutputAudioMode</code> .

Return Values

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

2.3.21.2 **IBMDSwitcherMixMinusOutput::GetAudioMode** method

The `GetAudioMode` method returns the current audio mode of the mix minus output.

Syntax

```
HRESULT GetAudioMode(BMDSwitcherMixMinusOutputAudioMode* audioMode)
```

Parameters

Name	Direction	Description
audioMode	out	The current audio mode as a <code>BMDSwitcherMixMinusOutputAudioMode</code> .

Return Values

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

2.3.21.3 **IBMDSwitcherMixMinusOutput::SetAudioMode** method

The `SetAudioMode` method sets the audio mode of the mix minus output.

Syntax

```
HRESULT SetAudioMode(BMDSwitcherMixMinusOutputAudioMode audioMode)
```

Parameters

Name	Direction	Description
audioMode	in	The desired audio mode in <code>BMDSwitcherMixMinusOutputAudioMode</code> .

Return Values

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

2.3.21.4 **IBMDSwitcherMixMinusOutput::HasMinusAudioInputId** method

The `HasMinusAudioInputId` method returns the current has minus audio input ID flag.

Syntax

```
HRESULT HasMinusAudioInputId(boolean* hasMinusAudioInputId)
```

Parameters

Name	Direction	Description
hasMinusAudioInputId	out	The current has minus audio input ID flag.

Return Values

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

2.3.21.5 **IBMDSwitcherMixMinusOutput::GetMinusAudioInputId** method

The `GetMinusAudioInputId` method gets the `BMDSwitcherAudioInputId` of the audio input that is subtracted when the output is in mix minus audio mode.

Syntax

```
HRESULT GetMinusAudioInputId(BMDSwitcherAudioInputId* audioInputId)
```

Parameters

Name	Direction	Description
audioInputId	out	The <code>BMDSwitcherAudioInputId</code> of the audio input used for mix minus.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The <code>audioInputId</code> parameter is invalid.
E_FAIL	No minus audio input assigned.

2.3.21.6 **IBMDSwitcherMixMinusOutput::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherMixMinusOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherMixMinusOutputCallback</code> object interface.

Return Values

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

2.3.21.7 IBMDSwitcherMixMinusOutput::RemoveCallback method

The RemoveCallback method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherMixMinusOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherMixMinusOutputCallback object interface.

Return Values

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

2.3.22 IBMDSwitcherMixMinusOutputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixMinusOutput	IID_IBMDSwitcherMixMinusOutput	An IBMDSwitcherMixMinusOutputCallback object interface is installed with IBMDSwitcherMixMinusOutput::AddCallback and removed with IBMDSwitcherMixMinusOutput::RemoveCallback.

Public Member Functions

Method	Description
Notify	Called when an event occurs.

2.3.22.1 IBMDSwitcherMixMinusOutputCallback::Notify method

The **Notify** method is called when **IBMDSwitcherMixMinusOutput** events occur, such as audio mode 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(IBMDSwitcherMixMinusOutputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherMixMinusOutputEventType that describes the type of event that occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

2.3.23 IBMDSwitcherSaveRecall Interface

The **IBMDSwitcherSaveRecall** object interface provides functionality for storing and clearing operating states.

Public Member Functions	
Method	Description
Save	Stores the current operating state to the switcher.
Clear	Clears a stored operating state from the switcher.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An IBMDSwitcherSaveRecall object interface can be obtained with IBMDSwitcher::QueryInterface .

2.3.23.1 IBMDSwitcherSaveRecall::Save method

The Save method stores the current operating state to the switcher's persistent memory.

Syntax

```
HRESULT Save(BMDSwitcherSaveRecallType type)
```

Parameters

Name	Direction	Description
type	in	The type of operating state to store.

Return Values

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

2.3.23.2 IBMDSwitcherSaveRecall::Clear method

The Clear method clears a stored operating state from the switcher's persistent memory.

Syntax

```
HRESULT Clear(BMDSwitcherSaveRecallType type)
```

Parameters

Name	Direction	Description
type	in	The type of operating state to clear.

Return Values

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

Section 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.

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**.

- **bmdSwitcherTransitionMixParametersEventTypeRateChanged**
The rate changed.

3.1.2 Dip Parameters Event Type

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

- **bmdSwitcherTransitionDipParametersEventTypeRateChanged**
The rate changed.
- **bmdSwitcherTransitionDipParametersEventTypeInputDipChanged**
The dip input changed.

3.1.3 Wipe Parameters Event Type

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

- **bmdSwitcherTransitionWipeParametersEventTypeRateChanged**
The rate changed.
- **bmdSwitcherTransitionWipeParametersEventTypePatternChanged**
The pattern changed.
- **bmdSwitcherTransitionWipeParametersEventTypeBorderSizeChanged**
The border size changed.
- **bmdSwitcherTransitionWipeParametersEventTypeInputBorderChanged**
The border input changed.
- **bmdSwitcherTransitionWipeParametersEventTypeSymmetryChanged**
The symmetry changed.
- **bmdSwitcherTransitionWipeParametersEventTypeSoftnessChanged**
The softness changed.
- **bmdSwitcherTransitionWipeParametersEventTypeHorizontalOffsetChanged**
The horizontal offset changed.
- **bmdSwitcherTransitionWipeParametersEventTypeVerticalOffsetChanged**
The vertical offset changed.
- **bmdSwitcherTransitionWipeParametersEventTypeReverseChanged**
The reverse flag changed.
- **bmdSwitcherTransitionWipeParametersEventTypeFlipFlopChanged**
The flip flop flag changed.

3.1.4 DVE Parameters Event Type

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

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

3.1.5 Stinger Parameters Event Type

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

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

3.1.6 Transition Parameters Event Type

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

- **bmdSwitcherTransitionParametersEventTypeTransitionStyleChanged**
The transition style changed.
- **bmdSwitcherTransitionParametersEventTypeNextTransitionStyleChanged**
The next transition style changed.
- **bmdSwitcherTransitionParametersEventTypeTransitionSelectionChanged**
The transition selection changed.
- **bmdSwitcherTransitionParametersEventTypeNextTransitionSelectionChanged**
The next transition selection changed.

3.1.7 Transition Style

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

- **bmdSwitcherTransitionStyleMix** Mix style.
- **bmdSwitcherTransitionStyleDip** Dip style.
- **bmdSwitcherTransitionStyleWipe** Wipe style.
- **bmdSwitcherTransitionStyleDVE** DVE style.
- **bmdSwitcherTransitionStyleStinger** 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.

- **bmdSwitcherTransitionSelectionBackground** Include background in transition.
- **bmdSwitcherTransitionSelectionKey1** Include key 1 in transition.
- **bmdSwitcherTransitionSelectionKey2** Include key 2 in transition.
- **bmdSwitcherTransitionSelectionKey3** Include key 3 in transition.
- **bmdSwitcherTransitionSelectionKey4** Include key 4 in transition.

3.1.9 DVE Transition Style

BMDSwitcherDVETransitionStyle enumerates the possible transition styles. This type is used by the **IBMDSwitcherTransitionDVEParameters** object interface.

- **bmdSwitcherDVETransitionStyleSwooshTopLeft**
Top left swoosh.
- **bmdSwitcherDVETransitionStyleSwooshTop**
Top swoosh.
- **bmdSwitcherDVETransitionStyleSwooshTopRight**
Top right swoosh.
- **bmdSwitcherDVETransitionStyleSwooshLeft**
Left swoosh.
- **bmdSwitcherDVETransitionStyleSwooshRight**
Right swoosh.

- **bmdSwitcherDVETransitionStyleSwooshBottomLeft**
Bottom left swoosh.
- **bmdSwitcherDVETransitionStyleSwooshBottom**
Bottom swoosh.
- **bmdSwitcherDVETransitionStyleSwooshBottomRight**
Bottom right swoosh.
- **bmdSwitcherDVETransitionStyleSpinCWTopLeft**
Top left clockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCWTopRight**
Top right clockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCWBottomLeft**
Bottom left clockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCWBottomRight**
Bottom right clockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCCWTopLeft**
Top left counterclockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCCWTopRight**
Top right counterclockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCCWBottomLeft**
Bottom left counterclockwise spin.
- **bmdSwitcherDVETransitionStyleSpinCCWBottomRight**
Bottom right counterclockwise spin.
- **bmdSwitcherDVETransitionStyleSqueezeTopLeft**
Top left squeeze.
- **bmdSwitcherDVETransitionStyleSqueezeTop**
Top squeeze.
- **bmdSwitcherDVETransitionStyleSqueezeTopRight**
Top right squeeze.
- **bmdSwitcherDVETransitionStyleSqueezeLeft**
Left squeeze.
- **bmdSwitcherDVETransitionStyleSqueezeRight**
Right squeeze.
- **bmdSwitcherDVETransitionStyleSqueezeBottomLeft**
Bottom left squeeze.

3.1.10 Stinger Transition Source

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

- **bmdSwitcherStingerTransitionSourceMediaPlayer1**
Media player 1.
- **bmdSwitcherStingerTransitionSourceMediaPlayer2**
Media player 2.
- **bmdSwitcherStingerTransitionSourceMediaPlayer3**
Media player 3.
- **bmdSwitcherStingerTransitionSourceMediaPlayer4**
Media player 4.
- **bmdSwitcherStingerTransitionSourceNone**
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
<code>IBMDSwitcherMixEffectBlock</code>	<code>IID_IBMDSwitcherMixEffectBlock</code>	An <code>IBMDSwitcherTransitionMixParameters</code> object interface can be obtained with <code>IBMDSwitcherMixEffectBlock::QueryInterface</code> .

Public Member Functions

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

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
<code>frames</code>	out	The current rate.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The frames parameter is invalid.

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 IBMDSwitcherTransitionMixParametersCallback object interface.

Return Values

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

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 <code>IBMDSwitcherTransitionMixParametersCallback</code> 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 <code>IBMDSwitcherTransitionMixParametersCallback</code> object interface is installed with <code>IBMDSwitcherTransitionMixParameters::AddCallback</code> and removed with <code>IBMDSwitcherTransitionMixParameters::RemoveCallback</code>

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(IBMDSwitcherTransitionMixParametersEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherTransitionMixParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

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 IBMDSwitcherTransitionDipParameters object interface can be obtained with IBMDSwitcherMixEffectBlock::QueryInterface .

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

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.

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.

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.

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.

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 <code>IBMDSwitcherTransitionDipParametersCallback</code> object interface.

Return Values

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

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 <code>IBMDSwitcherTransitionDipParametersCallback</code> 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
<code>IBMDSwitcherTransitionDipParameters</code>	<code>IID_IBMDSwitcherTransitionDipParameters</code>	An <code>IBMDSwitcherTransitionDipParametersCallback</code> object interface is installed with <code>IBMDSwitcherTransitionDipParameters::AddCallback</code> and removed with <code>IBMDSwitcherTransitionDipParameters::RemoveCallback</code>

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

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(IBMDSwitcherTransitionDipParametersEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherTransitionDipParametersEventType 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	IBMDSwitcherTransitionWipeParametersCallback object interface is installed with IBMDSwitcherTransitionWipeParameters::AddCallback and removed with IBMDSwitcherTransitionWipeParameters::RemoveCallback

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

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(IBMDSwitcherTransitionWipeParametersEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherTransitionWipeParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

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 IBMDSwitcherTransitionWipeParameters object interface can be obtained with IBMDSwitcherMixEffectBlock::QueryInterface .

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.

Public Member Functions	
Method	Description
SetSymmetry	Set the symmetry.
GetSoftness	Get the current softness.
SetSoftness	Set the softness.
GetHorizontalOffset	Get the current horizontal offset.
SetHorizontalOffset	Set the horizontal offset.
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.

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.

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.

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.

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.

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.

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.

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	92.323

Return Values

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

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.

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.

3.2.6.13 **IBMDSwitcherTransitionWipeParameters::GetHorizontalOffset** method

The `GetHorizontalOffset` method returns the current horizontal offset.

Syntax

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

Parameters

Name	Direction	Description
<code>hOffset</code>	out	The current horizontal offset.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>hOffset</code> 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
<code>hOffset</code>	in	The desired horizontal offset.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	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
<code>vOffset</code>	out	The current vertical offset.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>vOffset</code> 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.

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 IBMDSwitcherTransitionWipeParametersCallback object interface.

Return Values

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

3.2.6.22 IBMDSwitcherTransitionWipeParameters::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherTransitionWipeParametersCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherTransitionWipeParametersCallback object interface.

Return Values

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

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
<code>IBMDSwitcherMixEffectBlock</code>	<code>IID_IBMDSwitcherMixEffectBlock</code>	An <code>IBMDSwitcherTransitionDVEParameters</code> object interface can be obtained with <code>IBMDSwitcherMixEffectBlock::QueryInterface</code> .

Public Member Functions	
Method	Description
<code>GetRate</code>	Get the current rate.
<code>SetRate</code>	Set the rate.
<code>GetLogoRate</code>	Get the current logo rate.
<code>SetLogoRate</code>	Set the logo rate.
<code>GetReverse</code>	Get the current reverse flag.
<code>SetReverse</code>	Set the reverse flag.
<code>GetFlipFlop</code>	Get the current flip flop flag.
<code>SetFlipFlop</code>	Set the flip flop flag.
<code>GetStyle</code>	Get the current style.
<code>SetStyle</code>	Set the style.
<code>DoesSupportStyle</code>	Gets whether the connected device supports a given DVE transition style.
<code>GetNumSupportedStyles</code>	Gets the number of supported DVE transition style types.
<code>GetSupportedStyle</code>	Gets one or more supported DVE transition styles.
<code>GetInputFill</code>	Get the current fill input.
<code>SetInputFill</code>	Set the fill input.
<code>GetInputCut</code>	Get the current cut input.
<code>SetInputCut</code>	Set the cut input.
<code>GetFillInputAvailabilityMask</code>	Get the availability mask for the fill of this input.
<code>GetCutInputAvailabilityMask</code>	Get the availability mask for the cut of this input.
<code>GetEnableKey</code>	Get the current enable key.
<code>SetEnableKey</code>	Set the enable key.
<code>GetPreMultiplied</code>	Get the current pre-multiplied flag.
<code>SetPreMultiplied</code>	Set the pre-multiplied flag.
<code>GetClip</code>	Get the current clip value.
<code>SetClip</code>	Set the clip value.
<code>GetGain</code>	Get the current gain.
<code>SetGain</code>	Set the gain.
<code>GetInverse</code>	Get the current inverse flag.
<code>SetInverse</code>	Set the inverse flag.
<code>AddCallback</code>	Add a callback.

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.

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.

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.

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.

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.

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.

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.

3.2.7.11 **IBMDSwitcherTransitionDVEParameters::DoesSupportStyle** method

The **DoesSupportStyle** method determines if the connected device supports a given DVE transition style.

Syntax

```
HRESULT DoesSupportStyle(BMDSwitcherDVETransitionStyle style, BMDbool* supported)
```

Parameters

Name	Direction	Description
style	in	The DVE style to check.
supported	out	Boolean status of the requested DVE transition style support.

Return Values

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

3.2.7.12 **IBMDSwitcherTransitionDVEParameters::GetNumSupportedStyles** method

The **GetNumSupportedStyles** method determines the total number of supported DVE transition styles in the connected device.

Syntax

```
HRESULT GetNumSupportedStyles(uint32_t* numSupportedStyles)
```

Parameters

Name	Direction	Description
numSupportedStyles	out	Total number of DVE transition styles supported.

Return Values

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

3.2.7.13 IBMDSwitcherTransitionDVEParameters::GetSupportedStyles method

The `GetSupportedStyles` method retrieves a list of supported DVE transition styles supported by the connected device.

NOTE That it is the user application's responsibility to ensure the supportedStyles destination array is at least as big as `supportedStylesMaxCount`.

Syntax

```
HRESULT GetSupportedStyles(BMDSwitcherDVETransitionStyle* supportedStyles,
                          uint32_t supportedStylesMaxCount)
```

Parameters

Name	Direction	Description
<code>supportedStyles</code>	out	List of supported DVE transition styles.
<code>supportedStylesMaxCount</code>	in	Maximum number of supported DVE transition styles to retrieve.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The supportedStyles parameter is invalid.

3.2.7.14 IBMDSwitcherTransitionDVEParameters::GetInputFill method

The `GetInputFill` method returns the current fill input.

Syntax

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

Parameters

Name	Direction	Description
<code>input</code>	out	The current fill input.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The input parameter is invalid.

3.2.7.15 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.

3.2.7.16 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.

3.2.7.17 **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.

3.2.7.18 **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 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.

3.2.7.19 **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 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.

3.2.7.20 **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.

3.2.7.21 **IBMDSwitcherTransitionDVEParameters::SetEnableKey** method

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

Syntax

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

Parameters

Name	Direction	Description
<code>enableKey</code>	in	The desired <code>enableKey</code> flag.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	Failure.

3.2.7.22 **IBMDSwitcherTransitionDVEParameters::GetPreMultiplied** method

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

Syntax

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

Parameters

Name	Direction	Description
<code>enableKey</code>	in	The desired <code>enableKey</code> flag.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>preMultiplied</code> parameter is invalid.

3.2.7.23 **IBMDSwitcherTransitionDVEParameters::SetPreMultiplied** method

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

Syntax

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

Parameters

Name	Direction	Description
<code>preMultiplied</code>	in	The desired pre-multiplied flag.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

3.2.7.24 **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.

3.2.7.25 **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.

3.2.7.26 **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.

3.2.7.27 **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.

3.2.7.28 **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.

3.2.7.29 **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.

3.2.7.30 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 IBMDSwitcherTransitionDVEParametersCallback object interface.

Return Values

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

3.2.7.31 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 IBMDSwitcherTransitionDVEParametersCallback 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
<code>IBMDSwitcherTransitionDVEParametersCallback</code>	<code>IID_IBMDSwitcherTransitionDVEParameters</code>	An <code>IBMDSwitcherTransitionDVEParametersCallback</code> object interface is installed with <code>IBMDSwitcherTransitionDVEParameters::AddCallback</code> and removed with <code>IBMDSwitcherTransitionDVEParameters::RemoveCallback</code>

Public Member Functions	
Method	Description
<code>Notify</code>	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
<code>eventType</code>	in	<code>BMDSwitcherTransitionDVEParametersEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	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
<code>IBMDSwitcherMixEffectBlock</code>	<code>IID_IBMDSwitcherMixEffectBlock</code>	An <code>IBMDSwitcherTransitionStingerParameters</code> object interface can be obtained with <code>IBMDSwitcherMixEffectBlock::QueryInterface</code> .

Public Member Functions	
Method	Description
<code>GetSource</code>	Get the current source.
<code>SetSource</code>	Set the source.
<code>GetPreMultiplied</code>	Get the current pre-multiplied flag.
<code>SetPreMultiplied</code>	Set the pre-multiplied flag.
<code>GetClip</code>	Get the current clip value.
<code>SetClip</code>	Set the clip value.
<code>GetGain</code>	Get the current gain.
<code>SetGain</code>	Set the gain.
<code>GetInverse</code>	Get the current inverse flag.
<code>SetInverse</code>	Set the inverse flag.
<code>GetPreroll</code>	Get the current pre-roll.
<code>SetPreroll</code>	Set the pre-roll.
<code>GetClipDuration</code>	Get the current clip duration.
<code>SetClipDuration</code>	Set the clip duration.
<code>GetTriggerPoint</code>	Get the current trigger point.
<code>SetTriggerPoint</code>	Set the trigger point.
<code>GetMixRate</code>	Get the current mix rate.
<code>SetMixRate</code>	Set the mix rate.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

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.

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.

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.

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.

3.2.9.7 **IBMDSwitcherTransitionStingerParameters::GetGain** method

The `GetGain` method returns the current 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.

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.

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.

3.2.9.11 **IBMDSwitcherTransitionStingerParameters::GetPreroll** method

The `GetPreroll` method returns the current pre-roll.

Syntax

```
HRESULT GetPreroll(UINT32* 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 frames)
```

Parameters

Name	Direction	Description
frames	in	The desired pre-roll in frames.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

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.

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.

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 IBMDSwitcherTransitionStingerParametersCallback object interface.

Return Values

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

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 IBMDSwitcherTransitionStingerParametersCallback 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
<code>IBMDSwitcherTransitionStingerParameters</code>	<code>IID_IBMDSwitcherTransitionStingerParameters</code>	An <code>IBMDSwitcherTransitionStingerParametersCallback</code> object interface is installed with <code>IBMDSwitcherTransitionStingerParameters::AddCallback</code> and removed with <code>IBMDSwitcherTransitionStingerParameters::RemoveCallback</code>

Public Member Functions	
Method	Description
<code>Notify</code>	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
<code>eventType</code>	in	<code>BMDSwitcherTransitionStingerParametersEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

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 <code>IBMDSwitcherTransitionParameters</code> object interface can be obtained with <code>IBMDSwitcherMixEffectBlock::QueryInterface</code> .

Public Member Functions

Method	Description
<code>GetTransitionStyle</code>	Get the current transition style.
<code>GetNextTransitionStyle</code>	Get the next transition style.
<code>SetNextTransitionStyle</code>	Set the next transition style.
<code>GetTransitionSelection</code>	Get the current transition selection.
<code>SetNextTransitionSelection</code>	Set the next transition selection.
<code>GetNextTransitionSelection</code>	Get the next transition selection.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

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.

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.

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.

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.

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.

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 IBMDSwitcherTransitionParametersCallback object interface.

Return Values

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

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 IBMDSwitcherTransitionParametersCallback 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
<code>IBMDSwitcherTransitionParameters</code>	<code>IID_IBMDSwitcherTransitionParameters</code>	An <code>IBMDSwitcherTransitionParametersCallback</code> object interface is installed with <code>IBMDSwitcherTransitionParameters::AddCallback</code> and removed with <code>IBMDSwitcherTransitionParameters::RemoveCallback</code>

Public Member Functions

Method	Description
<code>Notify</code>	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
<code>eventType</code>	in	<code>BMDSwitcherTransitionParametersEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

Section 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**.
- 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:

- 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 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.
- 5 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.

4.2 Media Data Types

4.2.1 Switcher Pixel Format

BMDSwitcherPixelFormat enumerates the possible pixel formats for **IBMDSwitcherFrame**.

- **bmdSwitcherPixelFormat8BitARGB**
Four bytes per pixel, alpha, red, green, blue.
- **bmdSwitcherPixelFormat8BitXRGB**
Four bytes per pixel, padding, red, green, blue.
- **bmdSwitcherPixelFormat8BitYUV**
Four bytes per two pixels, cb, y0, cr, y1.
- **bmdSwitcherPixelFormat10BitYUVA**
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**.

- **bmdSwitcherMediaPlayerSourceTypeStill**
Still.
- **bmdSwitcherMediaPlayerSourceTypeClip**
Clip.

4.2.3 Media Pool Event Type

BMDSwitcherMediaPoolEventType enumerates the possible event types for **IBMDSwitcherStillsCallback** and **IBMDSwitcherClipCallback**.

- **bmdSwitcherMediaPoolEventTypeValidChanged**
The validity has changed.
- **bmdSwitcherMediaPoolEventTypeNameChanged**
The name has changed.
- **bmdSwitcherMediaPoolEventTypeHashChanged**
The hash has changed.
- **bmdSwitcherMediaPoolEventTypeAudioValidChanged**
The audio validity has changed.
- **bmdSwitcherMediaPoolEventTypeLockBusy**
All clients receive this when any client obtains a lock.
- **bmdSwitcherMediaPoolEventTypeLockIdle**
All clients receive this when no client has lock.
- **bmdSwitcherMediaPoolEventTypeTransferCompleted**
The transfer has completed.
- **bmdSwitcherMediaPoolEventTypeTransferCancelled**
The transfer has cancelled.
- **bmdSwitcherMediaPoolEventTypeTransferFailed**
The transfer has failed.
- **bmdSwitcherMediaPoolEventTypeAudioNameChanged**
The audio name has changed.
- **bmdSwitcherMediaPoolEventTypeAudioHashChanged**
The audio hash has changed.

4.2.4 Still Capture Event Type

BMDSwitcherStillCaptureEventType enumerates the possible event types for **IBMDSwitcherStillCaptureCallback**.

- **bmdSwitcherStillCaptureEventTypesAvailableChanged**
The availability of still capture 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 IBMDSwitcherMediaPlayerCallback object interface is installed with IBMDSwitcherMediaPlayer::AddCallback and removed with IBMDSwitcherMediaPlayer::RemoveCallback

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.

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.

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.

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	<code>IBMDSwitcher::CreateIterator</code> can return an <code>IBMDSwitcherMediaPlayerIterator</code> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <code>IBMDSwitcherMediaPlayer</code> 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	<code>IBMDSwitcherMediaPlayer</code> object interface or NULL when no more media players are available.

Return Values

Value	Description
S_FALSE	No more <code>IBMDSwitcherMediaPlayer</code> objects available.
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	<code>IBMDSwitcherMediaPlayerIterator::Next</code> returns an <code>IBMDSwitcherMediaPlayer</code> 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	<code>BMDSwitcherMediaPlayerSourceType</code> 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	BMDSwitcherMediaPlayerSourceType 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.

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 IBMDSwitcherMediaPlayerCallback 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 IBMDSwitcherMediaPlayerCallback 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	IBMDSwitcherMediaPool::CreateFrame returns an IBMDSwitcherFrame 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 object remains valid.

Return Values

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

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	IBMDSwitcherMediaPool::CreateAudio returns an IBMDSwitcherAudio 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 IBMDSwitcherLockCallback object interface is installed with IBMDSwitcherStills::Lock and removed with IBMDSwitcherStills::Unlock
IBMDSwitcherClip	IID_IBMDSwitcherClip	An IBMDSwitcherLockCallback object interface is installed with IBMDSwitcherClip::Lock and removed with IBMDSwitcherClip::Unlock

Public Member Functions	
Method	Description
Obtained	Called when the client receives a lock.

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 IBMDSwitcherStillsCallback object interface is installed with IBMDSwitcherStills::AddCallback and removed with IBMDSwitcherStills::RemoveCallback

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.

- **bmdSwitcherMediaPoolEventTypeTransferCompleted**
IBMDSwitcherStills::Upload and **IBMDSwitcherStills::Download**.
- **bmdSwitcherMediaPoolEventTypeTransferCancelled**
IBMDSwitcherStills::Upload only.
- **bmdSwitcherMediaPoolEventTypeTransferFailed**
BMDSwitcherStills::Upload 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	BMDSwitcherMediaPoolEventType that describes the type of event that has occurred.
frame	in	The IBMDSwitcherFrame that is being transferred. May be NULL .
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	IBMDSwitcherMediaPool::GetStills returns an IBMDSwitcherStills 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 IBMDSwitcherLockCallback 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 IBMDSwitcherLockCallback 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
index	in	Destination still index.
name	in	Destination still name.
frame	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.

IBMDSwitcherStillsCallback::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 IBMDSwitcherStillsCallback 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 <code>IBMDSwitcherStillsCallback</code> object interface.

Return Values

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

4.3.9 IBMDSwitcherStillCaptureCallback Interface

The `IBMDSwitcherStillCaptureCallback` object interface is a callback class containing methods that are called when an event occurs on an `IBMDSwitcherStillCapture` object.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherStillCapture</code>	<code>IID_IBMDSwitcherStillCapture</code>	An <code>IBMDSwitcherStillCaptureCallback</code> object interface is installed with <code>IBMDSwitcherStillCapture::AddCallback</code> and removed with <code>IBMDSwitcherStillCapture::RemoveCallback</code> .

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

4.3.9.1 IBMDSwitcherStillCaptureCallback::Notify method

The Notify method is called when **IBMDSwitcherStillCapture** 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(BMDSwitcherStillCaptureEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherStillCaptureEventType that describes the type of event that has occurred.

4.3.10 IBMDSwitcherStillCapture Interface

The **IBMDSwitcherStillCapture** object interface provides functionality for capturing the program output to media pool stills.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherStills	IID_IBMDSwitcherStills	An IBMDSwitcherStillCapture object interface can be obtained with IBMDSwitcherStills::QueryInterface .

Public Member Functions	
Method	Description
IsAvailable	Determines whether the switcher is currently able to capture the program output to a media pool still.
CaptureStill	Captures the current program output frame as a media pool still.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

4.3.10.1 **IBMDSwitcherStillCapture::IsAvailable** method

The `IsAvailable` method is used to determine whether the switcher is currently able to capture the program output frame. Capture requires the media pool to have at least one empty still.

Syntax

```
HRESULT IsAvailable(Boolean* available)
```

Parameters

Name	Direction	Description
available	out	Boolean that indicates whether still capture is currently available.

Return Values

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

4.3.10.2 **IBMDSwitcherStillCapture::CaptureStill** method

The `CaptureStill` method captures the current program output frame and stores it in the first empty still in the media pool.

Syntax

```
HRESULT CaptureStill()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen when there are no empty stills in the media pool.

4.3.10.3 **IBMDSwitcherStillCapture::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherStillCaptureCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherStillCaptureCallback object interface.

Return Values

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

4.3.10.4 IBMDSwitcherStillCapture::RemoveCallback method

The RemoveCallback method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherStillCaptureCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherStillCaptureCallback object interface.

Return Values

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

4.3.11 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 IBMDSwitcherClipCallback object interface is installed with IBMDSwitcherClip::AddCallback and removed with IBMDSwitcherClip::RemoveCallback

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

4.3.11.1 **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:

- **bmdSwitcherMediaPoolEventTypeTransferCompleted**
IBMDSwitcherClip::Upload and **IBMDSwitcherClip::Download**.
- **bmdSwitcherMediaPoolEventTypeTransferCancelled**
IBMDSwitcherClip::Upload only.
- **bmdSwitcherMediaPoolEventTypeTransferFailed**
IBMDSwitcherClip::Upload only.

The audio is set during the following transfer events, otherwise it is NULL:

- **bmdSwitcherMediaPoolEventTypeTransferCompleted**
IBMDSwitcherClip::UploadAudio and **IBMDSwitcherClip::DownloadAudio**.
- **bmdSwitcherMediaPoolEventTypeTransferCancelled**
IBMDSwitcherClip::UploadAudio only.
- **bmdSwitcherMediaPoolEventTypeTransferFailed**
IBMDSwitcherClip::UploadAudio 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)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherMediaPoolEventType that describes the type of event that has occurred.
frame	in	The IBMDSwitcherFrame 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 IBMDSwitcherAudio that is being transferred. May be NULL.
clipIndex	in	Specifies the clip for the eventType.

Return Values

Value	Description
S_OK	Success.

4.3.12 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	IBMDSwitcherMediaPool::GetClip returns an IBMDSwitcherClip 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.
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.

4.3.12.1 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.

4.3.12.2 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.

4.3.12.3 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.

4.3.12.4 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.

4.3.12.5 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.

4.3.12.6 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.

4.3.12.7 **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.

4.3.12.8 **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.

4.3.12.9 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.

4.3.12.10 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.

4.3.12.11 **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.

4.3.12.12 **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.

4.3.12.13 **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.

4.3.12.14 **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.

4.3.12.15 **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.

4.3.12.16 **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 IBMDSwitcherLockCallback object interface.

Return Values

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

4.3.12.17 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 IBMDSwitcherLockCallback object interface.

Return Values

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

4.3.12.18 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.

4.3.12.19 **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.

4.3.12.20 **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(string name, IBMDSwitcherAudio* audio)
```

Parameters

Name	Direction	Description
name	in	Descriptive name of the audio.
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.

4.3.12.21 **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.

4.3.12.22 **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.

4.3.12.23 **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.

4.3.12.24 **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 IBMDSwitcherClipCallback object interface.

Return Values

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

4.3.12.25 **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 IBMDSwitcherClipCallback object interface.

Return Values

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

4.3.13 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 IBMDSwitcherMediaPoolCallback object interface is installed with IBMDSwitcherMediaPool::AddCallback and removed with IBMDSwitcherMediaPool::RemoveCallback

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.

4.3.13.1 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.

4.3.13.2 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.

4.3.14 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 IBMDSwitcherMediaPool object interface can be obtained with IBMDSwitcher::QueryInterface .

Public Member Functions	
Method	Description
GetStills	Gets the IBMDSwitcherStills object interface.
GetClip	Gets a IBMDSwitcherClip object interface.
GetClipCount	Gets the number of clips.
CreateFrame	Creates an IBMDSwitcherFrame object.
CreateAudio	Creates an IBMDSwitcherAudio 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.
Clear	clears all stills, clips and clip audio
DoesVideoModeChangeClearMediaPool	Determines if changing video standard will clear the media pool.
AddCallback	Adds a media pool callback.
RemoveCallback	Removes a media pool callback.

4.3.14.1 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.

4.3.14.2 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 <code>clipIndex</code> parameter is invalid.
S_OK	Success.

4.3.14.3 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 <code>clipCount</code> parameter is invalid.
S_OK	Success.

4.3.14.4 **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 BMDSwitcherPixelFormat 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.

4.3.14.5 **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.

4.3.14.6 **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.

4.3.14.7 **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.

4.3.14.8 **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.

4.3.14.9 **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.

4.3.14.10 **IBMDSwitcherMediaPool::DoesVideoModeChangeClearMediaPool** method

The **DoesVideoModeChangeClearMediaPool** method determines if changing to the specified video standard will clear the media pool.

Syntax

```
HRESULT DoesVideoModeChangeClearMediaPool(IBMDSwitcherVideoMode videoMode,  
Boolean* clear)
```

Parameters

Name	Direction	Description
videoMode	in	The video standard.
clear	out	Boolean value that is true if changing to the video standard will clear the media pool.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The videoMode is not a valid video standard.
E_POINTER	The required parameter is invalid.

4.3.14.11 **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 IBMDSwitcherMediaPoolCallback object interface.

Return Values

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

4.3.14.12 **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 IBMDSwitcherMediaPoolCallback object interface.

Return Values

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

Section 5 — Keyers

Any form of a keyer available in our switchers use these API components to perform keying.

5.1 Key Data Types

5.1.1 Key Type

BMDSwitcherKeyType enumerates the possible key types, used by **IBMDSwitcherKey** object interface.

- **bmdSwitcherKeyTypeLuma**
Luminance key.
- **bmdSwitcherKeyTypeChroma**
Chroma key.
- **bmdSwitcherKeyTypePattern**
Pattern key.
- **bmdSwitcherKeyTypeDVE**
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.

- **bmdSwitcherFlyKeyFrameFull**
Full screen.
- **bmdSwitcherFlyKeyFrameInfinityCentreOfKey**
Infinity at centre of key.
- **bmdSwitcherFlyKeyFrameInfinityTopLeft**
Infinity at top left of screen.
- **bmdSwitcherFlyKeyFrameInfinityTop**
Infinity at top centre of screen.
- **bmdSwitcherFlyKeyFrameInfinityTopRight**
Infinity at top right of screen.
- **bmdSwitcherFlyKeyFrameInfinityLeft**
Infinity at mid left of screen.
- **bmdSwitcherFlyKeyFrameInfinityCentre**
Infinity at centre of screen.
- **bmdSwitcherFlyKeyFrameInfinityRight**
Infinity at mid right of screen.
- **bmdSwitcherFlyKeyFrameInfinityBottomLeft**
Infinity at bottom left of screen.
- **bmdSwitcherFlyKeyFrameInfinityBottom**
Infinity at bottom centre of screen.
- **bmdSwitcherFlyKeyFrameInfinityBottomRight**
Infinity at bottom right of screen.
- **bmdSwitcherFlyKeyFrameA**
User-defined key frame A.
- **bmdSwitcherFlyKeyFrameB**
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.

- **bmdSwitcherBorderBevelOptionNone**
No bevel.
- **bmdSwitcherBorderBevelOptionInOut**
Both inner and outer bevel.
- **bmdSwitcherBorderBevelOptionIn**
Inner bevel only.
- **bmdSwitcherBorderBevelOptionOut**
Outer bevel only.

5.1.4 Key Event Type

BMDSwitcherKeyEventType enumerates the possible event types for **IBMDSwitcherKeyCallback**.

- **bmdSwitcherKeyEventTypeTypeChanged**
The type changed.
- **bmdSwitcherKeyEventTypeInputCutChanged**
The cut input source changed.
- **bmdSwitcherKeyEventTypeInputFillChanged**
The fill input source changed.
- **bmdSwitcherKeyEventTypeOnAirChanged**
The on-air flag changed.
- **bmdSwitcherKeyEventTypeCanBeDVEKeyChanged**
The can-be-DVE flag changed.
- **bmdSwitcherKeyEventTypeMaskedChanged**
The masked flag changed.
- **bmdSwitcherKeyEventTypeMaskTopChanged**
The mask top value changed.
- **bmdSwitcherKeyEventTypeMaskBottomChanged**
The mask bottom value changed.
- **bmdSwitcherKeyEventTypeMaskLeftChanged**
The mask left value changed.
- **bmdSwitcherKeyEventTypeMaskRightChanged**
The mask right value changed.

5.1.5 Luminance Key Parameters Event Type

BMDSwitcherKeyLumaParametersEventType enumerates the possible event types for **IBMDSwitcherKeyLumaParametersCallback**.

- **bmdSwitcherKeyLumaParametersEventTypePreMultipliedChanged**
The pre-multiplied flag changed.
- **bmdSwitcherKeyLumaParametersEventTypeClipChanged**
The clip value changed.
- **bmdSwitcherKeyLumaParametersEventTypeGainChanged**
The gain value changed.
- **bmdSwitcherKeyLumaParametersEventTypeInverseChanged**
The inverse flag changed.

5.1.6 Chroma Key Parameters Event Type

BMDSwitcherKeyChromaParametersEventType enumerates the possible event types for **IBMDSwitcherKeyChromaParametersCallback**.

- **bmdSwitcherKeyChromaParametersEventTypeHueChanged**
The hue value changed.
- **bmdSwitcherKeyChromaParametersEventTypeGainChanged**
The gain value changed.
- **bmdSwitcherKeyChromaParametersEventTypeYSuppressChanged**
The y-suppress value changed.
- **bmdSwitcherKeyChromaParametersEventTypeLiftChanged**
The lift value changed.
- **bmdSwitcherKeyChromaParametersEventTypeNarrowChanged**
The narrow flag changed.

5.1.7 Pattern Key Parameters Event Type

BMDSwitcherKeyAdvancedChromaParametersEventType enumerates the possible event types for **IBMDSwitcherKeyAdvancedChromaParametersCallback**.

- **bmdSwitcherKeyAdvancedChromaParametersEventTypeForegroundLevelChanged**
The key adjustment foreground level value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeBackgroundLevelChanged**
The key adjustment background level value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeKeyEdgeChanged**
The key adjustment key edge value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeSpillSuppressChanged**
The chroma correction spill suppress value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeFlareSuppressChanged**
The chroma correction flare suppress value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeBrightnessChanged**
The color adjustment brightness value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeContrastChanged**
The color adjustment contrast value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeSaturationChanged**
The color adjustment saturation value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeRedChanged**
The color adjustment red value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeGreenChanged**
The color adjustment green value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeBlueChanged**
The color adjustment blue value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeSamplingModeEnabled Changed**
The sampling mode enabled flag changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypePreviewEnabledChanged**
The preview enabled flag changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeCursorXPositionChanged**
The cursor X position value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeCursorYPositionChanged**
The cursor Y position value changed.

- **bmdSwitcherKeyAdvancedChromaParametersEventTypeCursorSizeChanged**
The cursor size value changed.
- **bmdSwitcherKeyAdvancedChromaParametersEventTypeSampledColorChanged**
The sampled color value changed.

5.1.8 Pattern Key Parameters Event Type

BMDSwitcherKeyPatternParametersEventType enumerates the possible event types for **IBMDSwitcherKeyPatternParametersCallback**.

- **bmdSwitcherKeyPatternParametersEventTypePatternChanged**
The pattern style changed.
- **bmdSwitcherKeyPatternParametersEventTypeSizeChanged**
The size value changed.
- **bmdSwitcherKeyPatternParametersEventTypeSymmetryChanged**
The symmetry value changed.
- **bmdSwitcherKeyPatternParametersEventTypeSoftnessChanged**
The softness value changed.
- **bmdSwitcherKeyPatternParametersEventTypeHorizontalOffsetChanged**
The horizontal offset changed.
- **bmdSwitcherKeyPatternParametersEventTypeVerticalOffsetChanged**
The vertical offset changed.
- **bmdSwitcherKeyPatternParametersEventTypeInverseChanged**
The inverse flag changed.

5.1.9 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.

5.1.10 Fly Key Parameters Event Type

BMDSwitcherKeyFlyParametersEventType enumerates the possible event types for **IBMDSwitcherKeyFlyParametersCallback**.

- **bmdSwitcherKeyFlyParametersEventTypeFlyChanged**
The fly flag changed.
- **bmdSwitcherKeyFlyParametersEventTypeCanFlyChanged**
The can-fly flag changed.
- **bmdSwitcherKeyFlyParametersEventTypeRateChanged**
The rate value changed.
- **bmdSwitcherKeyFlyParametersEventTypeSizeXChanged**
The size x value changed.
- **bmdSwitcherKeyFlyParametersEventTypeSizeYChanged**
The size y value changed.
- **bmdSwitcherKeyFlyParametersEventTypePositionXChanged**
The position x value changed.
- **bmdSwitcherKeyFlyParametersEventTypePositionYChanged**
The position y value changed.
- **bmdSwitcherKeyFlyParametersEventTypeRotationChanged**
The rotation value changed.
- **bmdSwitcherKeyFlyParametersEventTypeIsKeyFrameStoredChanged**
The is-key-frame-stored flag changed.
- **bmdSwitcherKeyFlyParametersEventTypeIsAtKeyFramesChanged**
The is-at-key-frames status changed.
- **bmdSwitcherKeyFlyParametersEventTypeIsRunningChanged**
The is-running status changed.

5.1.11 Downstream Key Event Type

BMDSwitcherDownstreamKeyEventType enumerates the possible event types for **IBMDSwitcherDownstreamKeyCallback**.

- **bmdSwitcherDownstreamKeyEventTypeInputCutChanged**
The cut input source changed.
- **bmdSwitcherDownstreamKeyEventTypeInputFillChanged**
The fill input source changed.
- **bmdSwitcherDownstreamKeyEventTypeTieChanged**
The tie flag changed.
- **bmdSwitcherDownstreamKeyEventTypeRateChanged**
The rate value changed.
- **bmdSwitcherDownstreamKeyEventTypeOnAirChanged**
The on-air flag changed.
- **bmdSwitcherDownstreamKeyEventTypesTransitioningChanged**
The is-transitioning flag changed.
- **bmdSwitcherDownstreamKeyEventTypesAutoTransitioningChanged**
The is-auto-transitioning flag changed.
- **bmdSwitcherDownstreamKeyEventTypesTransitionTowardsOnAirChanged**
The is-transitioning-towards-on-air flag changed.
- **bmdSwitcherDownstreamKeyEventTypeFramesRemainingChanged**
The frames remaining value changed.
- **bmdSwitcherDownstreamKeyEventTypePreMultipliedChanged**
The pre-multiplied flag changed.
- **bmdSwitcherDownstreamKeyEventTypeClipChanged**
The clip value changed.
- **bmdSwitcherDownstreamKeyEventTypeGainChanged**
The gain value changed.
- **bmdSwitcherDownstreamKeyEventTypeInverseChanged**
The inverse flag changed.
- **bmdSwitcherDownstreamKeyEventTypeMaskedChanged**
The masked flag changed.
- **bmdSwitcherDownstreamKeyEventTypeMaskTopChanged**
The mask top value changed.
- **bmdSwitcherDownstreamKeyEventTypeMaskBottomChanged**
The mask bottom value changed.
- **bmdSwitcherDownstreamKeyEventTypeMaskLeftChanged**
The mask left value changed.
- **bmdSwitcherDownstreamKeyEventTypeMaskRightChanged**
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 **Createrlterator** method. Pass **IID_IBMDSwitcherKeylterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherMixEffectBlock	IID_IBMDSwitcherMixEffectBlock	IBMDSwitcherMixEffectBlock::Createrlterator can return an IBMDSwitcherKeylterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherKey object interface.

5.2.1.1 IBMDSwitcherKeylterator::Next method

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

Syntax

```
HRESULT Next (IBMDSwitcherKey* key);
```

Parameters

Name	Direction	Description
key	out	IBMDSwitcherKey object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherKey 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 IBMDSwitcherKey object will be returned after a successful call to IBMDSwitcherKeyIterator::Next method.

Public Member Functions	
Method	Description
DoesSupportAdvancedChroma	The DoesSupportAdvancedChroma method determines if advanced chroma key is supported by the switcher.
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.
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 BMDSwitcherTransitionSelection bit mask for this key.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

5.2.2.1 IBMDSwitcherKey::DoesSupportAdvancedChroma method

The DoesSupportAdvancedChroma method determines if advanced chroma key is supported by the switcher.

Syntax

```
HRESULT DoesSupportAdvancedChroma (boolean* supportsAdvancedChroma)
```

Parameters

Name	Direction	Description
supportsAdvancedChroma	out	Boolean value describing whether the advanced chroma key is supported by the switcher.

5.2.2.2 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.

5.2.2.3 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.

5.2.2.4 **IBMDSwitcherKey::GetInputCut** method

The `GetInputCut` method returns the selected cut input source.

Syntax

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

Parameters

Name	Direction	Description
inputId	out	BMDSwitcherInputId of the selected cut input source.

Return Values

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

5.2.2.5 **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 BMDSwitcherInputId.

Return Values

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

5.2.2.6 **IBMDSwitcherKey::GetInputFill** method

The **GetInputFill** method returns the selected fill input source.

Syntax

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

Parameters

Name	Direction	Description
inputId	out	BMDSwitcherInputId of the selected fill input source.

Return Values

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

5.2.2.7 **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 BMDSwitcherInputId .

Return Values

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

5.2.2.8 **IBMDSwitcherKey::GetFillInputAvailabilityMask** method

The **GetFillInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for fill inputs available to this key. The input availability property 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.

5.2.2.9 **IBMDSwitcherKey::GetCutInputAvailabilityMask** method

The **GetCutInputAvailabilityMask** method returns the corresponding **BMDSwitcherInputAvailability** bit mask value for cut inputs available to this key. The input availability property 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.

5.2.2.10 **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.

5.2.2.11 **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.

5.2.2.12 **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.

5.2.2.13 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.

5.2.2.14 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.

5.2.2.15 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.

5.2.2.16 **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.

5.2.2.17 **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.

5.2.2.18 **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.

5.2.2.19 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.

5.2.2.20 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.

5.2.2.21 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.

5.2.2.22 **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.

5.2.2.23 **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.

5.2.2.24 **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	BMDSwitcherTransitionSelection bit mask.

Return Values

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

5.2.2.25 **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 IBMDSwitcherKeyCallback object interface.

Return Values

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

5.2.2.26 **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 IBMDSwitcherKeyCallback 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 IBMDSwitcherKeyCallback object interface is installed with IBMDSwitcherKey::AddCallback and removed with IBMDSwitcherKey::RemoveCallback .

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	BMDSwitcherKeyEventType 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
<code>IBMDSwitcherKey</code>	<code>IID_IBMDSwitcherKey</code>	An <code>IBMDSwitcherKeyLumaParameters</code> object interface can be obtained with <code>IBMDSwitcherKey::QueryInterface</code> .

Public Member Functions

Method	Description
<code>GetPreMultiplied</code>	Get the current pre-multiplied flag.
<code>SetPreMultiplied</code>	Set pre-multiplied flag.
<code>GetClip</code>	Get the current clip value.
<code>SetClip</code>	Set the clip value.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set gain value.
<code>GetInverse</code>	Get the current inverse flag.
<code>SetInverse</code>	Set the inverse flag.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	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
<code>preMultiplied</code>	out	The current pre-multiplied flag.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>preMultiplied</code> 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 IBMDSwitcherKeyLumaParametersCallback 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 IBMDSwitcherKeyLumaParametersCallback 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 IBMDSwitcherKeyLumaParametersCallback object interface is installed with IBMDSwitcherKeyLumaParameters::AddCallback and removed with IBMDSwitcherKeyLumaParameters::RemoveCallback

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	BMDSwitcherKeyLumaParametersEventType 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.

If a switcher is capable of using advanced chroma key, then this interface will not be available. Only if `IBMDSwitcherKey::DoesSupportAdvancedChroma` returns false, does the switcher support this `IBMDSwitcherKeyChromaParameters` interface.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An <code>IBMDSwitcherKeyChromaParameters</code> object interface can be obtained with <code>IBMDSwitcherKey::QueryInterface</code> .

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 IBMDSwitcherKeyChromaParametersCallback 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 IBMDSwitcherKeyChromaParametersCallback 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 IBMDSwitcherKeyChromaParametersCallback object interface is installed with IBMDSwitcherKeyChromaParameters::AddCallback and removed with IBMDSwitcherKeyChromaParameters::RemoveCallback

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	BMDSwitcherKeyChromaParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8 IBMDSwitcherKeyAdvancedChromaParameters Interface

The **IBMDSwitcherKeyAdvancedChromaParameters** object interface is used for manipulating settings specific to the advanced chroma type key.

Advanced chroma key is an improved version of chroma key and is not available on all models of switchers. Use **IBMDSwitcherKey::DoesSupportAdvancedChroma** to determine if a switcher supports the **IBMDSwitcherKeyAdvancedChromaParameters** interface.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKey	IID_IBMDSwitcherKey	An IBMDSwitcherKeyAdvancedChromaParameters object interface can be obtained with IBMDSwitcherKey::QueryInterface .

Public Member Functions	
Method	Description
GetForegroundLevel	Get the current key adjustment foreground level value.
SetForegroundLevel	Set the key adjustment foreground level value.
GetBackgroundLevel	Get the current key adjustment background level value.
SetBackgroundLevel	Set the key adjustment background level value.
GetKeyEdge	Get the current key adjustment key edge value.
SetKeyEdge	Set the key adjustment key edge value.
GetSpillSuppress	Get the current chroma correction spill suppress value.
SetSpillSuppress	Set the chroma correction spill suppress value.
GetFlareSuppress	Get the current chroma correction flare suppress value.
SetFlareSuppress	Set the chroma correction flare suppress value.
GetBrightness	Get the current color adjustment brightness value.
SetBrightness	Set the color adjustment brightness value.
GetContrast	Get the current color adjustment contrast value.
SetContrast	Set the color adjustment contrast value.
GetSaturation	Get the current color adjustment saturation value.
SetSaturation	Set the color adjustment saturation value.
GetRed	Get the current color adjustment red value.
SetRed	Set the color adjustment red value.
GetGreen	Get the current color adjustment green value.
SetGreen	Set the color adjustment green value.
GetBlue	Get the current color adjustment blue value.
SetBlue	Set the color adjustment blue value.

Public Member Functions	
Method	Description
GetSamplingModeEnabled	Get the current sampling mode enabled flag.
SetSamplingModeEnabled	Set the sampling mode enabled flag.
GetPreviewEnabled	Get the current preview enabled flag.
SetPreviewEnabled	Set the preview enabled flag.
GetCursorXPosition	Get the current cursor x position.
SetCursorXPosition	Set the cursor x position.
GetCursorYPosition	Get the current cursor y position.
SetCursorYPosition	Set the cursor y position.
GetCursorSize	Get the current cursor size.
SetCursorSize	Set the cursor size.
GetSampledColor	Get the current sampled color.
SetSampledColor	Set the sampled color.
ResetKeyAdjustments	Reset key adjustment properties to default values.
ResetChromaCorrection	Reset chroma correction properties to default values.
ResetColorAdjustments	Reset color adjustment properties to default values.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

5.2.8.1 **IBMDSwitcherKeyAdvancedChromaParameters:: GetForegroundLevel** method

The **GetForegroundLevel** method gets the current key adjustment foreground level value.

Syntax

```
HRESULT GetForegroundLevel (double* level);
```

Parameters

Name	Direction	Description
level	out	The current key adjustment foreground level value.

Return Values

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

5.2.8.2 **IBMDSwitcherKeyAdvancedChromaParameters::SetForegroundLevel** method

The **SetForegroundLevel** method sets the key adjustment foreground level value.

Syntax

```
HRESULT SetForegroundLevel (double level);
```

Parameters

Name	Direction	Description
level	in	The desired key adjustment foreground level value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.3 **IBMDSwitcherKeyAdvancedChromaParameters::GetBackgroundLevel** method

The **GetBackgroundLevel** method gets the current key adjustment background level value.

Syntax

```
HRESULT GetBackgroundLevel (double* level);
```

Parameters

Name	Direction	Description
level	out	The current key adjustment background level value.

Return Values

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

5.2.8.4 **IBMDSwitcherKeyAdvancedChromaParameters::SetBackgroundLevel** method

The **SetBackgroundLevel** method sets the key adjustment background level value.

Syntax

```
HRESULT SetBackgroundLevel (double level);
```

Parameters

Name	Direction	Description
level	in	The desired key adjustment background level value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.5 **IBMDSwitcherKeyAdvancedChromaParameters::GetKeyEdge** method

The **GetKeyEdge** method gets the current key adjustment key edge value.

Syntax

```
HRESULT GetKeyEdge (double* keyEdge);
```

Parameters

Name	Direction	Description
keyEdge	out	The current key adjustment key edge value.

Return Values

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

5.2.8.6 **IBMDSwitcherKeyAdvancedChromaParameters::SetKeyEdge** method

The **SetKeyEdge** method sets the key adjustment key edge value.

Syntax

```
HRESULT SetKeyEdge (double keyEdge);
```

Parameters

Name	Direction	Description
keyEdge	in	The desired key adjustment key edge value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.7 **IBMDSwitcherKeyAdvancedChromaParameters::GetSpillSuppress** method

The **GetSpillSuppress** method gets the current chroma correction spill suppress value.

Syntax

```
HRESULT GetSpillSuppress (double* spillSuppress);
```

Parameters

Name	Direction	Description
spillSuppress	out	The current chroma correction spill suppress value.

Return Values

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

5.2.8.8 **IBMDSwitcherKeyAdvancedChromaParameters::SetSpillSuppress** method

The **SetSpillSuppress** method sets the chroma correction spill suppress value.

Syntax

```
HRESULT SetSpillSuppress (double spillSuppress);
```

Parameters

Name	Direction	Description
spillSuppress	in	The desired chroma correction spill suppress value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.9 **IBMDSwitcherKeyAdvancedChromaParameters::GetFlareSuppress** method

The **GetFlareSuppress** method gets the current chroma correction flare suppress value.

Syntax

```
HRESULT GetFlareSuppress (double* flareSuppress);
```

Parameters

Name	Direction	Description
flareSuppress	out	The current chroma correction flare suppress value.

Return Values

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

5.2.8.10 **IBMDSwitcherKeyAdvancedChromaParameters::SetFlareSuppress** method

The **SetFlareSuppress** method sets the chroma correction flare suppress value.

Syntax

```
HRESULT SetFlareSuppress (double flareSuppress);
```

Parameters

Name	Direction	Description
flareSuppress	in	The desired chroma correction flare suppress value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.11 **IBMDSwitcherKeyAdvancedChromaParameters::GetBrightness** method

The `GetBrightness` method gets the current color adjustment brightness value.

Syntax

```
HRESULT GetBrightness (double* brightness);
```

Parameters

Name	Direction	Description
brightness	out	The current color adjustment brightness value.

Return Values

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

5.2.8.12 **IBMDSwitcherKeyAdvancedChromaParameters::SetBrightness** method

The `SetBrightness` method sets the color adjustment brightness value.

Syntax

```
HRESULT SetBrightness (double brightness);
```

Parameters

Name	Direction	Description
brightness	in	The desired color adjustment brightness value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.13 **IBMDSwitcherKeyAdvancedChromaParameters::GetContrast** method

The `GetContrast` method gets the current color adjustment contrast value.

Syntax

```
HRESULT GetContrast (double* contrast);
```

Parameters

Name	Direction	Description
contrast	out	The current color adjustment contrast value.

Return Values

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

5.2.8.14 **IBMDSwitcherKeyAdvancedChromaParameters::SetContrast** method

The **SetContrast** method sets the color adjustment contrast value.

Syntax

```
HRESULT SetContrast (double contrast);
```

Parameters

Name	Direction	Description
contrast	in	The desired color adjustment contrast value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.15 **IBMDSwitcherKeyAdvancedChromaParameters::GetSaturation** method

The **GetSaturation** method gets the current color adjustment saturation value.

Syntax

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

Parameters

Name	Direction	Description
saturation	out	The current color adjustment saturation value.

Return Values

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

5.2.8.16 **IBMDSwitcherKeyAdvancedChromaParameters::SetSaturation** method

The **SetSaturation** method sets the color adjustment saturation value.

Syntax

```
HRESULT SetSaturation (double saturation);
```

Parameters

Name	Direction	Description
saturation	in	The desired color adjustment saturation value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.17 **IBMDSwitcherKeyAdvancedChromaParameters::GetRed** method

The **GetRed** method gets the current color adjustment red value.

Syntax

```
HRESULT GetRed (double* red);
```

Parameters

Name	Direction	Description
red	out	The current color adjustment red value.

Return Values

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

5.2.8.18 **IBMDSwitcherKeyAdvancedChromaParameters::SetRed** method

The **SetRed** method sets the color adjustment red value.

Syntax

```
HRESULT SetRed (double red);
```

Parameters

Name	Direction	Description
red	in	The desired color adjustment red value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.19 **IBMDSwitcherKeyAdvancedChromaParameters::GetGreen** method

The **GetGreen** method gets the current color adjustment green value.

Syntax

```
HRESULT GetGreen (double* green);
```

Parameters

Name	Direction	Description
green	out	The current color adjustment green value.

Return Values

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

5.2.8.20 **IBMDSwitcherKeyAdvancedChromaParameters::SetGreen** method

The **SetGreen** method sets the color adjustment green value.

Syntax

```
HRESULT SetGreen (double green);
```

Parameters

Name	Direction	Description
green	in	The desired color adjustment green value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.21 **IBMDSwitcherKeyAdvancedChromaParameters::GetBlue** method

The **GetBlue** method gets the current color adjustment blue value.

Syntax

```
HRESULT GetBlue (double* blue);
```

Parameters

Name	Direction	Description
blue	out	The current color adjustment blue value.

Return Values

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

5.2.8.22 **IBMDSwitcherKeyAdvancedChromaParameters::SetBlue** method

The **SetBlue** method sets the color adjustment blue value.

Syntax

```
HRESULT SetBlue (double blue);
```

Parameters

Name	Direction	Description
blue	in	The desired color adjustment blue value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.23 **IBMDSwitcherKeyAdvancedChromaParameters::GetSamplingModeEnabled** method

The **GetSamplingModeEnabled** method gets the current sampling mode enabled flag.

Syntax

```
HRESULT GetSamplingModeEnabled (boolean* enabled);
```

Parameters

Name	Direction	Description
enabled	out	The current sampling mode enabled flag.

Return Values

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

5.2.8.24 **IBMDSwitcherKeyAdvancedChromaParameters::SetSamplingModeEnabled** method

The **SetSamplingModeEnabled** method sets the sampling mode enabled flag.

Syntax

```
HRESULT SetSamplingModeEnabled (boolean enabled);
```

Parameters

Name	Direction	Description
enabled	in	The desired sampling mode enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.25 **IBMDSwitcherKeyAdvancedChromaParameters::GetPreviewEnabled** method

The `GetPreviewEnabled` method gets the current preview enabled flag.

Syntax

```
HRESULT GetPreviewEnabled (boolean* enabled);
```

Parameters

Name	Direction	Description
enabled	out	The enabled parameter is invalid.

Return Values

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

5.2.8.26 **IBMDSwitcherKeyAdvancedChromaParameters::SetPreviewEnabled** method

The `SetPreviewEnabled` method sets the preview enabled flag.

Syntax

```
HRESULT SetPreviewEnabled (boolean enabled);
```

Parameters

Name	Direction	Description
enabled	in	The desired preview enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.27 **IBMDSwitcherKeyAdvancedChromaParameters::GetCursorXPosition** method

The `GetCursorXPosition` method gets the current cursor x position value.

Syntax

```
HRESULT GetCursorXPosition (double* position);
```

Parameters

Name	Direction	Description
position	out	The current cursor x position value.

Return Values

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

5.2.8.28 **IBMDSwitcherKeyAdvancedChromaParameters::SetCursorXPosition** method

The `SetCursorXPosition` method sets the cursor x position value.

Syntax

```
HRESULT SetCursorXPosition (double position);
```

Parameters

Name	Direction	Description
position	in	The desired cursor x position value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.29 **IBMDSwitcherKeyAdvancedChromaParameters::GetCursorYPosition** method

The `GetCursorYPosition` method gets the current cursor y position value.

Syntax

```
HRESULT GetCursorYPosition (double* position);
```

Parameters

Name	Direction	Description
position	out	The current cursor y position value.

Return Values

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

5.2.8.30 **IBMDSwitcherKeyAdvancedChromaParameters::SetCursorYPosition** method

The `SetCursorYPosition` method sets the cursor y position value.

Syntax

```
HRESULT SetCursorYPosition (double position);
```

Parameters

Name	Direction	Description
position	in	The desired cursor y position value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.31 **IBMDSwitcherKeyAdvancedChromaParameters::GetCursorSize** method

The `GetCursorSize` method gets the current cursor size value.

Syntax

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

Parameters

Name	Direction	Description
size	out	The current cursor size value.

Return Values

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

5.2.8.32 **IBMDSwitcherKeyAdvancedChromaParameters::SetCursorSize** method

The `SetCursorSize` method sets the cursor size value.

Syntax

```
HRESULT SetCursorSize (double size);
```

Parameters

Name	Direction	Description
size	in	The desired cursor size value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.33 **IBMDSwitcherKeyAdvancedChromaParameters::GetSampledColor** method

The **GetSampledColor** method gets the current sampled color value. The sampled color is in YCbCr format.

Syntax

```
HRESULT GetCursorSize (double* y, double* cb, double* cr);
```

Parameters

Name	Direction	Description
y	out	The current sampled color y value.
cb	out	The current sampled color cb value.
cr	out	The current sampled color cr value.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The y, cb, or cr parameter is invalid.

5.2.8.34 **IBMDSwitcherKeyAdvancedChromaParameters::SetSampledColor** method

The **SetSampledColor** method sets the sampled color value. The sampled color is in YCbCr format.

Syntax

```
HRESULT SetSampledColor (double y, double cb, double, cr);
```

Parameters

Name	Direction	Description
y	in	The desired sampled color y value.
cb	in	The desired sampled color cb value.
cr	in	The desired sampled color cr value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.35 **IBMDSwitcherKeyAdvancedChromaParameters::ResetKeyAdjustments** method

The **ResetKeyAdjustments** method resets the key adjustment properties to default values. This includes foreground level, background level, and key edge values.

Syntax

```
HRESULT ResetKeyAdjustments (void);
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.36 **IBMDSwitcherKeyAdvancedChromaParameters::ResetChromaCorrection** method

The **ResetChromaCorrection** method resets the chroma correction properties to default values. This includes spill suppress, and flare suppress values.

Syntax

```
HRESULT ResetChromaCorrection (void);
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.37 **IBMDSwitcherKeyAdvancedChromaParameters::ResetColorAdjustments** method

The **ResetColorAdjustments** method resets the color adjustment properties to default values. This includes brightness, contrast, saturation, red, green, and blue values.

Syntax

```
HRESULT ResetColorAdjustments (void);
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.8.38 **IBMDSwitcherKeyAdvancedChromaParameters::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherKeyAdvancedChromaParameters** object. Pass an object implementing the **IBMDSwitcherKeyAdvancedChromaParametersCallback** interface to receive callbacks.

Adding a new callback will not affect previously added callbacks.

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherKeyAdvancedChromaParametersCallback object interface.

Return Values

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

5.2.8.39 **IBMDSwitcherKeyAdvancedChromaParameters::RemoveCallback** method

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

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherKeyAdvancedChromaParametersCallback object interface.

Return Values

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

5.2.9 IBMDSwitcherKeyAdvancedChromaParametersCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherKeyAdvancedChromaParameters	IID_IBMDSwitcherKeyAdvancedChromaParameters	An IBMDSwitcherKeyAdvancedChromaParametersCallback object interface is installed with IBMDSwitcherKeyAdvancedChromaParameters::AddCallback and removed with IBMDSwitcherKeyAdvancedChromaParameters::RemoveCallback .

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

5.2.9.1 IBMDSwitcherKeyAdvancedChromaParametersCallback::Notify method

The **Notify** method is called when **IBMDSwitcherKeyAdvancedChromaParameters** 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 (IBMDSwitcherKeyAdvancedChromaParametersEventType eventType);
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherKeyAdvancedChromaParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.10 IBMDSwitcherKeyPatternParameters Interface

The `IBMDSwitcherKeyPatternParameters` object interface is used for manipulating settings specific to the pattern type key.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherKey</code>	<code>IID_IBMDSwitcherKey</code>	An <code>IBMDSwitcherKeyPatternParameters</code> object interface can be obtained with <code>IBMDSwitcherKey::QueryInterface</code> .

Public Member Functions

Method	Description
<code>GetPattern</code>	Get the current pattern style.
<code>SetPattern</code>	Set the pattern style.
<code>GetSize</code>	Get the current size value.
<code>SetSize</code>	Set the size value.
<code>GetSymmetry</code>	Get the current symmetry value.
<code>SetSymmetry</code>	Set the symmetry value.
<code>GetSoftness</code>	Get the current softness value.
<code>SetSoftness</code>	Set the softness value.
<code>GetHorizontalOffset</code>	Get the current horizontal offset.
<code>SetHorizontalOffset</code>	Set the horizontal offset.
<code>GetVerticalOffset</code>	Get the current vertical offset.
<code>SetVerticalOffset</code>	Set the vertical offset.
<code>GetInverse</code>	Get the current inverse flag.
<code>SetInverse</code>	Set the inverse flag.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

5.2.10.1 IBMDSwitcherKeyPatternParameters::GetPattern method

The `GetPattern` method gets the current pattern style.

Syntax

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

Parameters

Name	Direction	Description
<code>pattern</code>	out	The current pattern style of <code>BMDSwitcherPatternStyle</code> .

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The pattern parameter is invalid.

5.2.10.2 **IBMDSwitcherKeyPatternParameters::SetPattern** method

The **SetPattern** method sets the pattern style.

Syntax

```
HRESULT SetPattern (IBMDSwitcherPatternStyle pattern);
```

Parameters

Name	Direction	Description
pattern	in	The desired IBMDSwitcherPatternStyle pattern style.

Return Values

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

5.2.10.3 **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.

5.2.10.4 **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.

5.2.10.5 **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.

5.2.10.6 **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.

5.2.10.7 **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.

5.2.10.8 **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.

5.2.10.9 **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.

5.2.10.10 **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.

5.2.10.11 **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.

5.2.10.12 **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.

5.2.10.13 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.

5.2.10.14 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.

5.2.10.15 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 IBMDSwitcherKeyPatternParametersCallback object interface.

Return Values

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

5.2.10.16 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 IBMDSwitcherKeyPatternParametersCallback object interface.

Return Values

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

5.2.11 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 IBMDSwitcherKeyPatternParametersCallback object interface is installed with IBMDSwitcherKeyPatternParameters::AddCallback and removed with IBMDSwitcherKeyPatternParameters::RemoveCallback

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

5.2.11.1 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	BMDSwitcherKeyPatternParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.12 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 IBMDSwitcherKeyDVEParameters object interface can be obtained with IBMDSwitcherKey::QueryInterface .

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.
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.

Public Member Functions	
Method	Description
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.
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.

5.2.12.1 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.

5.2.12.2 **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.

5.2.12.3 **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.

5.2.12.4 **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.

5.2.12.5 **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.

5.2.12.6 **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.

5.2.12.7 **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.

5.2.12.8 **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.

5.2.12.9 **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 BMDSwitcherBorderBevelOption .

Return Values

Value	Description
S_OK	Success.
E_POINTER	The bevelOption parameter is invalid.
E_UNEXPECTED	Unexpected error occurred.

5.2.12.10 **IBMDSwitcherKeyDVEParameters::SetBorderBevel** method

The **SetBorderBevel** method sets the border bevel option.

Syntax

```
HRESULT SetBorderBevel (IBMDSwitcherBorderBevelOption bevelOption);
```

Parameters

Name	Direction	Description
bevelOption	in	The desired bevel option of IBMDSwitcherBorderBevelOption .

Return Values

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

5.2.12.11 **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.

5.2.12.12 **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.

5.2.12.13 **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.

5.2.12.14 **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.

5.2.12.15 **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_FAIL	Failure.

5.2.12.16 **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.

5.2.12.17 **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.

5.2.12.18 **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.

5.2.12.19 **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.

5.2.12.20 **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.

5.2.12.21 **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.

5.2.12.22 **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.

5.2.12.23 **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.

5.2.12.24 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.

5.2.12.25 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.

5.2.12.26 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.

5.2.12.27 **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.

5.2.12.28 **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.

5.2.12.29 **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.

5.2.12.30 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.

5.2.12.31 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.

5.2.12.32 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.

5.2.12.33 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.

5.2.12.34 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.

5.2.12.35 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.

5.2.12.36 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.

5.2.12.37 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.

5.2.12.38 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.

5.2.12.39 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.

5.2.12.40 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.

5.2.12.41 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.

5.2.12.42 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 IBMDSwitcherKeyDVEParametersCallback object interface.

Return Values

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

5.2.12.43 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 IBMDSwitcherKeyDVEParametersCallback object interface.

Return Values

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

5.2.13 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 <code>IBMDSwitcherKeyDVEParametersCallback</code> object interface is installed with <code>IBMDSwitcherKeyDVEParameters::AddCallback</code> and removed with <code>IBMDSwitcherKeyDVEParameters::RemoveCallback</code>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

5.2.13.1 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	<code>BMDSwitcherKeyDVEParametersEventType</code> that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.14 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 IBMDSwitcherKeyFlyParameters object interface can be obtained with IBMDSwitcherKey::QueryInterface .

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.
GetCanScaleUp	Gets whether the Fly Key size x and size y values can be greater than 1.0.
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.
GetRotation	Get the current rotation value.
SetRotation	Set the rotation value.
GetCanRotate	Gets whether the Fly Key supports rotation.
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.

5.2.14.1 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.

5.2.14.2 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.

5.2.14.3 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.

5.2.14.4 **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.

5.2.14.5 **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.

5.2.14.6 **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.

5.2.14.7 **IBMDSwitcherKeyFlyParameters::SetSizeX** method

The **SetSizeX** method sets the size x value. The flying size is a multiple of the original key size.

Note: On some switchers the maximum size x value is 1.0. The **GetCanScaleUp** method can be used to determine whether the switcher supports Fly Key size x values greater than 1.0.

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.

5.2.14.8 **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.

5.2.14.9 **IBMDSwitcherKeyFlyParameters::SetSizeY** method

The **SetSizeY** method sets the size y value. The flying size is a multiple of the original key size.

Note: On some switchers the maximum size y value is 1.0. The **GetCanScaleUp** method can be used to determine whether the switcher supports Fly Key size y values greater than 1.0.

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.

5.2.14.10 **IBMDSwitcherKeyFlyParameters::GetCanScaleUp** method

The **GetCanScaleUp** method is used to check whether the switcher supports Fly Key size x and size y values greater than 1.0.

Syntax

```
HRESULT GetCanScaleUp (boolean* canScaleUp);
```

Parameters

Name	Direction	Description
canScaleUp	out	A Boolean value indicating whether the switcher supports Fly Key size x and size y values greater than 1.0.

Return Values

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

5.2.14.11 **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.

5.2.14.12 **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.

5.2.14.13 **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.

5.2.14.14 **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.

5.2.14.15 **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.

5.2.14.16 **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.

5.2.14.17 **IBMDSwitcherKeyFlyParameters::GetCanRotate** method

The **GetCanRotate** method determines whether the current Fly Key supports rotation via the **SetRotation** method.

Syntax

```
HRESULT GetCanRotate (bool* canRotate);
```

Parameters

Name	Direction	Description
canRotate	out	The rotation support of the current Fly Key.

Return Values

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

5.2.14.18 **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.

5.2.14.19 **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.

5.2.14.20 **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.

5.2.14.21 **BMDSwitcherKeyFlyParameters::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 BMDSwitcherFlyKeyFrame 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.

5.2.14.22 **BMDSwitcherKeyFlyParameters::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.

5.2.14.23 **BMDSwitcherKeyFlyParameters::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.

5.2.14.24 **BMDSwitcherKeyFlyParameters::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.

5.2.14.25 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	IBMDSwitcherKeyFlyKeyFrameParameters object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The keyFrame parameter is invalid.
E_POINTER	The keyFrameParameters parameter is invalid.

5.2.14.26 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.

5.2.14.27 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 IBMDSwitcherKeyFlyParametersCallback object interface.

Return Values

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

5.2.14.28 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 IBMDSwitcherKeyFlyParametersCallback object interface.

Return Values

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

5.2.15 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 <code>IBMDSwitcherKeyFlyParametersCallback</code> object interface is installed with <code>IBMDSwitcherKeyFlyParameters::AddCallback</code> and removed with <code>IBMDSwitcherKeyFlyParameters::RemoveCallback</code>

Public Member Functions

Method	Description
Notify	Called when an event occurs.

5.2.15.1 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	<code>BMDSwitcherKeyFlyParametersEventType</code> that describes the type of event that has occurred.
keyFrame	in	This parameter is only valid when eventType is <code>bmdSwitcherKeyFlyParametersEventTypes KeyFrameStoredChanged</code> , it specifies the changed key frame.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.16 IBMDSwitcherKeyFlyKeyFrameParametersInterface

The `IBMDSwitcherKeyFlyKeyFrameParameters` object interface provides access to individual key frame parameters.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherKeyFlyParameters</code>	<code>IID_IBMDSwitcherKeyFlyParameters</code>	An <code>IBMDSwitcherKeyFlyKeyFrameParameters</code> object interface can be obtained from <code>IBMDSwitcherKeyFlyParameters::GetKeyFrameParameters</code> .

Public Member Functions	
Method	Description
<code>GetSizeX</code>	Get the size x value.
<code>SetSizeX</code>	Set the size x value.
<code>GetSizeY</code>	Get the size y value.
<code>SetSizeY</code>	Set the size y value.
<code>GetCanScaleUp</code>	Gets whether the Fly Key Key Frame size x and size y values can be greater than 1.0.
<code>GetPositionX</code>	Get the position x value.
<code>SetPositionX</code>	Set the position x value.
<code>GetPositionY</code>	Get the position y value.
<code>SetPositionY</code>	Set the position y value.
<code>GetRotation</code>	Get the rotation value.
<code>SetRotation</code>	Set the rotation value.
<code>GetCanRotate</code>	Gets whether the Fly Key Key Frame supports rotation.
<code>GetBorderWidthOut</code>	Get the border outer width value.
<code>SetBorderWidthOut</code>	Set the border outer width value.
<code>GetBorderWidthIn</code>	Get the border inner width value.
<code>SetBorderWidthIn</code>	Set the border inner width value.
<code>GetBorderSoftnessOut</code>	Get the border outer softness value.
<code>SetBorderSoftnessOut</code>	Set the border outer softness value.
<code>GetBorderSoftnessIn</code>	Get the border inner softness value.
<code>SetBorderSoftnessIn</code>	Set the border inner softness value.
<code>GetBorderBevelSoftness</code>	Get the border bevel softness value.
<code>SetBorderBevelSoftness</code>	Set the border bevel softness value.
<code>GetBorderBevelPosition</code>	Get the border bevel position value.
<code>SetBorderBevelPosition</code>	Set the border bevel position value.
<code>GetBorderOpacity</code>	Get the border opacity value.
<code>SetBorderOpacity</code>	Set the border opacity value.
<code>GetBorderHue</code>	Get the border hue value.
<code>SetBorderHue</code>	Set the border hue value.

Public Member Functions	
Method	Description
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.
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.

5.2.16.1 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.

5.2.16.2 **IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeX** method

The **SetSizeX** method sets the size x value.

Note: On some switchers the maximum size x value is 1.0. The **GetCanScaleUp** method can be used to determine whether the switcher supports Fly Key Key Frame size x values greater than 1.0.

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.

5.2.16.3 **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.

5.2.16.4 **IBMDSwitcherKeyFlyKeyFrameParameters::SetSizeY** method

The **SetSizeY** method sets the size y value.

Note: On some switchers the maximum size y value is 1.0. The **GetCanScaleUp** method can be used to determine whether the switcher supports Fly Key Key Frame size y values greater than 1.0.

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.

5.2.16.5 **IBMDSwitcherKeyFlyKeyFrameParameters::GetCanScaleUp** method

The **GetCanScaleUp** method is used to check whether the switcher supports Fly Key Key Frame size x and size y values greater than 1.0.

Syntax

```
HRESULT GetCanScaleUp (boolean* canScaleUp);
```

Parameters

Name	Direction	Description
canScaleUp	out	A Boolean value indicating whether the switcher supports Fly Key Key Frame size x and size y values greater than 1.0.

Return Values

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

5.2.16.6 **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.

5.2.16.7 **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.

5.2.16.8 **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.

5.2.16.9 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.

5.2.16.10 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.

5.2.16.11 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.

5.2.16.12 **IBMDSwitcherKeyFlyKeyFrameParameters::GetCanRotate** method

The `GetCanRotate` method determines whether the Fly Key Key Frame supports rotation via the `SetRotation` method.

Syntax

```
HRESULT GetCanRotate (bool* canRotate);
```

Parameters

Name	Direction	Description
canRotate	out	The rotation support of the Fly Key Key Frame.

Return Values

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

5.2.16.13 **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.

5.2.16.14 **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.

5.2.16.15 **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.

5.2.16.16 **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.

5.2.16.17 **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.

5.2.16.18 **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.

5.2.16.19 **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.

5.2.16.20 **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.

5.2.16.21 **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.

5.2.16.22 **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.

5.2.16.23 **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.

5.2.16.24 **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.

5.2.16.25 **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.

5.2.16.26 **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.

5.2.16.27 **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.

5.2.16.28 **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.

5.2.16.29 **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.

5.2.16.30 **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.

5.2.16.31 **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.

5.2.16.32 **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.

5.2.16.33 **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.

5.2.16.34 **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.

5.2.16.35 **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.

5.2.16.36 **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.

5.2.16.37 **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.

5.2.16.38 **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.

5.2.16.39 **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.

5.2.16.40 **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.

5.2.16.41 **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.

5.2.16.42 **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.

5.2.16.43 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.

5.2.16.44 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.

5.2.16.45 **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 IBMDSwitcherKeyFlyKeyFrameParametersCallback object interface.

Return Values

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

5.2.16.46 **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 IBMDSwitcherKeyFlyKeyFrameParametersCallback object interface.

Return Values

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

5.2.17 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_IBMDSwitcherKeyFlyKeyFrameParameters	An IBMDSwitcherKeyFlyKeyFrameParametersCallback object interface is installed with IBMDSwitcherKeyFlyKeyFrameParameters::AddCallback and removed with IBMDSwitcherKeyFlyKeyFrameParameters::RemoveCallback

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

5.2.17.1 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	BMDSwitcherKeyFlyKeyFrameParametersEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.18 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 **Createliterator** method. Pass **IID_IBMDSwitcherDownstreamKeylterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::Createliterator can return an IBMDSwitcherDownstreamKeylterator object interface.

Public Member Functions

Method	Description
Next	Returns a pointer to the next IBMDSwitcherDownstreamKey object interface.

5.2.18.1 IBMDSwitcherDownstreamKeylterator::Next method

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

Syntax

```
HRESULT Next (IBMDSwitcherDownstreamKey** downstreamKey);
```

Parameters

Name	Direction	Description
downstreamKey	out	IBMDSwitcherDownstreamKey object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherDownstreamKey objects available.
E_POINTER	The downstreamKey parameter is invalid.

5.2.19 IBMDSwitcherDownstreamKey Interface

The `IBMDSwitcherDownstreamKey` object interface is used for managing the settings of a downstream key.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherDownstreamKeyIterator</code>	<code>IID_IBMDSwitcherDownstreamKeyIterator</code>	An <code>IBMDSwitcherDownstreamKey</code> object will be returned after a successful call to <code>IBMDSwitcherDownstreamKeyIterator::Next</code> method.

Public Member Functions	
Method	Description
<code>GetInputCut</code>	Get the current cut input source.
<code>SetInputCut</code>	Set the cut input source.
<code>GetInputFill</code>	Get the current fill input source.
<code>SetInputFill</code>	Set the fill input source.
<code>GetFillInputAvailabilityMask</code>	Get the availability mask for the fill of this input.
<code>GetCutInputAvailabilityMask</code>	Get the availability mask for the cut of this input.
<code>GetTie</code>	Get the current tie flag.
<code>SetTie</code>	Set the tie flag.
<code>GetRate</code>	Get the current rate value.
<code>SetRate</code>	Set the rate value.
<code>GetOnAir</code>	Get the current on-air flag.
<code>SetOnAir</code>	Set the on-air flag.
<code>PerformAutoTransition</code>	Perform an auto-transition.
<code>PerformAutoTransitionInDirection</code>	Perform an auto-transition in a particular direction.
<code>IsTransitioning</code>	Determines if this downstream key is transitioning.
<code>IsAutoTransitioning</code>	Determines if this downstream key is auto-transitioning.
<code>IsTransitionTowardsOnAir</code>	Determines which direction this downstream key is transitioning.
<code>GetFramesRemaining</code>	Get the number of frames remaining in the transition.
<code>GetPreMultiplied</code>	Get the current pre-multiplied flag.
<code>SetPreMultiplied</code>	Set the pre-multiplied flag.
<code>GetClip</code>	Get the current clip value.
<code>SetClip</code>	Set the clip value.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set the gain value.
<code>GetInverse</code>	Get the current inverse flag.
<code>SetInverse</code>	Set the inverse flag.
<code>GetMasked</code>	Get the current masked flag.

Public Member Functions	
Method	Description
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.

5.2.19.1 **BMDSwitcherDownstreamKey::GetInputCut** method

The **GetInputCut** method returns the selected cut input source.

Syntax

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

Parameters

Name	Direction	Description
inputId	out	BMDSwitcherInputId of the selected cut input source.

Return Values

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

5.2.19.2 **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 BMDSwitcherInputId .

Return Values

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

5.2.19.3 **IBMDSwitcherDownstreamKey::GetInputFill** method

The **GetInputFill** method returns the selected fill input source.

Syntax

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

Parameters

Name	Direction	Description
inputId	out	BMDSwitcherInputId of the selected fill input source.

Return Values

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

5.2.19.4 **IBMDSwitcherDownstreamKey::SetInputFill** method

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

Syntax

```
HRESULT SetInputFill (IBMDSwitcherInputId inputId);
```

Parameters

Name	Direction	Description
inputId	in	The desired fill input source's IBMDSwitcherInputId .

Return Values

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

5.2.19.5 **IBMDSwitcherDownstreamKey::GetFillInputAvailabilityMask** method

The **GetFillInputAvailabilityMask** method returns the corresponding **IBMDSwitcherInputAvailability** bit mask value for fill inputs available to this downstream key. The input availability property 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 (IBMDSwitcherInputAvailability* mask);
```

Parameters

Name	Direction	Description
mask	out	IBMDSwitcherInputAvailability bit mask.

Return Values

Value	Description
S_OK	Success.
E_POINTER	Invalid mask parameter.

5.2.19.6 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 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.

5.2.19.7 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.

5.2.19.8 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.

5.2.19.9 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.

5.2.19.10 **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.

5.2.19.11 **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.

5.2.19.12 **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.

5.2.19.13 **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.

5.2.19.14 **IBMDSwitcherDownstreamKey::PerformAutoTransitionInDirection** method

The **PerformAutoTransitionInDirection** method performs an auto-transition in the specified direction, either towards on-air or away from on-air.

Syntax

```
HRESULT PerformAutoTransitionInDirection(boolean towardsOnAir)
```

Parameters

Name	Direction	Description
towardsOnAir	in	The desired direction.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

5.2.19.15 **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.

5.2.19.16 **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.

5.2.19.17 **IBMDSwitcherDownstreamKey::IsTransitionTowardsOnAir** method

The `IsTransitionTowardsOnAir` method returns whether this downstream key is transitioning towards or away from on-air.

Syntax

```
HRESULT IsTransitionTowardsOnAir (boolean* isTransitionTowardsOnAir)
```

Parameters

Name	Direction	Description
isTransitionTowardsOnAir	out	Boolean status of whether it is transitioning towards on-air.

Return Values

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

5.2.19.18 **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.

5.2.19.19 **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.

5.2.19.20 **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.

5.2.19.21 **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.

5.2.19.22 **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.

5.2.19.23 **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.

5.2.19.24 **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.

5.2.19.25 **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.

5.2.19.26 **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.

5.2.19.27 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.

5.2.19.28 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.

5.2.19.29 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.

5.2.19.30 **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.

5.2.19.31 **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.

5.2.19.32 **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.

5.2.19.33 **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.

5.2.19.34 **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.

5.2.19.35 **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.

5.2.19.36 **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.

5.2.19.37 **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.

5.2.19.38 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 IBMDSwitcherDownstreamKeyCallback object interface.

Return Values

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

5.2.19.39 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 IBMDSwitcherDownstreamKeyCallback object interface.

Return Values

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

5.2.20 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 IBMDSwitcherDownstreamKeyCallback object interface is installed with IBMDSwitcherDownstreamKey::AddCallback and removed with IBMDSwitcherDownstreamKey::RemoveCallback

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

5.2.20.1 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	BMDSwitcherDownstreamKeyEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 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 the **IBMDSwitcherSuperSourceBox** object interface.

- **bmdSwitcherSuperSourceBoxEventTypeInputSourceChanged**
The source input changed.
- **bmdSwitcherSuperSourceBoxEventTypePositionXChanged**
The x position changed.
- **bmdSwitcherSuperSourceBoxEventTypePositionYChanged**
The y position changed.
- **bmdSwitcherSuperSourceBoxEventTypeSizeChanged**
The size changed.
- **bmdSwitcherSuperSourceBoxEventTypeCroppedChanged**
The cropped flag changed.
- **bmdSwitcherSuperSourceBoxEventTypeCropTopChanged**
The top crop value changed.
- **bmdSwitcherSuperSourceBoxEventTypeCropBottomChanged**
The bottom crop value changed.
- **bmdSwitcherSuperSourceBoxEventTypeCropLeftChanged**
The left crop value changed.
- **bmdSwitcherSuperSourceBoxEventTypeCropRightChanged**
The right crop value changed.

6.1.2 SuperSource Border Event Type

BMDSwitcherSuperSourceBorderEventType enumerates the possible event types for the **IBMDSwitcherSuperSourceBorder** object interface.

- **bmdSwitcherSuperSourceBorderEventTypeEnabledChanged**
The border enabled flag changed.
- **bmdSwitcherSuperSourceBorderEventTypeBevelChanged**
The border bevel changed.
- **bmdSwitcherSuperSourceBorderEventTypeWidthOutChanged**
The border outer width changed.
- **bmdSwitcherSuperSourceBorderEventTypeWidthInChanged**
The border inner width changed.
- **bmdSwitcherSuperSourceBorderEventTypeSoftnessOutChanged**
The border outer softness changed.
- **bmdSwitcherSuperSourceBorderEventTypeSoftnessInChanged**
The border inner softness changed.
- **bmdSwitcherSuperSourceBorderEventTypeBevelSoftnessChanged**
The border bevel softness changed.
- **bmdSwitcherSuperSourceBorderEventTypeBevelPositionChanged**
The border bevel position changed.

- **bmdSwitcherSuperSourceBorderEventTypeHueChanged**
The border hue changed.
- **bmdSwitcherSuperSourceBorderEventTypeSaturationChanged**
The border saturation changed.
- **bmdSwitcherSuperSourceBorderEventTypeLumaChanged**
The border luminescence changed.
- **bmdSwitcherSuperSourceBorderEventTypeLightSourceDirectionChanged**
The border light source direction changed.
- **bmdSwitcherSuperSourceBorderEventTypeLightSourceAltitudeChanged**
The border light source altitude changed.

6.1.3 SuperSource Box Border Event Type

BMDSwitcherSuperSourceBoxBorderEventType enumerates the possible event types for the **IBMDSwitcherSuperSourceBoxBorder** object interface.

- **bmdSwitcherSuperSourceBoxBorderEventTypeEnabledChanged**
The border enabled flag changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthOutHorizontalChanged**
The border outer horizontal width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthOutVerticalChanged**
The border outer vertical width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthInLeftChanged**
The border inner left width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthInRightChanged**
The border inner right width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthInTopChanged**
The border inner top width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeWidthInBottomChanged**
The border inner bottom width changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeHueChanged**
The border hue changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeSaturationChanged**
The border saturation changed.
- **bmdSwitcherSuperSourceBoxBorderEventTypeLumaChanged**
The border luminescence changed.

6.1.4 SuperSource Input Event Type

BMDSwitcherInputSuperSourceEventType enumerates the possible event types for the **IBMDSwitcherInputSuperSource** object interface.

- **bmdSwitcherInputSuperSourceEventTypeInputFillChanged**
The fill input changed.
- **bmdSwitcherInputSuperSourceEventTypeInputCutChanged**
The cut input changed.
- **bmdSwitcherInputSuperSourceEventTypeArtOptionChanged**
The art option changed.
- **bmdSwitcherInputSuperSourceEventTypePreMultipliedChanged**
The pre-multiplied flag changed.
- **bmdSwitcherInputSuperSourceEventTypeClipChanged**
The clip value changed.

- **bmdSwitcherInputSuperSourceEventTypeGainChanged**
The gain changed.
- **bmdSwitcherInputSuperSourceEventTypeInverseChanged**
The inverse flag changed.

6.1.5 SuperSource Art Option

BMDSwitcherSuperSourceArtOption enumerates the possible supersource art options, used by the **IBMDSwitcherInputSuperSource** object interface.

- **bmdSwitcherSuperSourceArtOptionBackground**
Places art in the background.
- **bmdSwitcherSuperSourceArtOptionForeground**
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 IBMDSwitcherInputSuperSource object interface can be obtained with IBMDSwitcherInput::QueryInterface .

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.
SetGain	Set the art gain.
GetInverse	Get the current art inverse flag.

Public Member Functions	
Method	Description
SetInverse	Set the art inverse flag.
SupportsBorder	Determine if the SuperSource supports the display of borders.
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 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 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(IBMDSwitcherSuperSourceArtOption 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::SupportsBorder** method

The **SupportsBorder** method is used to determine if the switcher supports the display of borders on the SuperSource.

Syntax

```
HRESULT SupportsBorder(Boolean* supportsBorder)
```

Parameters

Name	Direction	Description
supportsBorder	out	Boolean value indicating whether borders are supported by the SuperSource.

Return Values

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

6.2.1.18 **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 IBMDSwitcherInputSuperSourceCallback object interface.

Return Values

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

6.2.1.19 **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 IBMDSwitcherInputSuperSourceCallback object interface.

Return Values

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

6.2.1.20 IBMDSwitcherInputSuperSource::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	out	Pointer to returned 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.

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 <code>IBMDSwitcherInputSuperSourceCallback</code> object interface is installed with <code>IBMDSwitcherInputSuperSource::AddCallback</code> and removed with <code>IBMDSwitcherInputSuperSource::RemoveCallback</code>

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(IBMDSwitcherInputSuperSourceEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherInputSuperSourceEventType 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	IBMDSwitcherInputSuperSource::CreateIterator can return an IBMDSwitcherSuperSourceBoxIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherSuperSourceBox 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	IBMDSwitcherSuperSourceBox object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherSuperSourceBox 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
IBMDSwitcherSuperSourceBox Iterator	IID_ IBMDSwitcherSuperSourceBox Iterator	An IBMDSwitcherSuperSourceBox object will be returned after a successful call to IBMDSwitcherSuperSourceBoxIterator::Next 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.
GetCropBottom	Get the bottom crop value.

Public Member Functions	
Method	Description
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 <code>BMDSwitcherInputId</code> .

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 <code>BMDSwitcherInputId</code> .

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 **IBMDSwitcherInputAvailability** bit mask value for this supersource box. The input availability property 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 source for this supersource box.

Syntax

```
HRESULT GetInputAvailabilityMask(IBMDSwitcherInputAvailability* mask)
```

Parameters

Name	Direction	Description
mask	out	IBMDSwitcherInputAvailability 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 IBMDSwitcherSuperSourceBoxCallback 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 <code>IBMDSwitcherSuperSourceBoxCallback</code> 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
<code>IBMDSwitcherSuperSourceBox</code>	<code>IID_IBMDSwitcherSuperSourceBox</code>	An <code>IBMDSwitcherSuperSourceBoxCallback</code> object interface is installed with <code>IBMDSwitcherSuperSourceBox::AddCallback</code> and removed with <code>IBMDSwitcherSuperSourceBox::RemoveCallback</code>

Public Member Functions

Method	Description
<code>Notify</code>	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(IBMDSwitcherSuperSourceBoxEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherSuperSourceBoxEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.6 IBMDSwitcherSuperSourceBorder Interface

The **IBMDSwitcherSuperSourceBorder** object interface is used for manipulating supersource border settings for all of the supersources boxes. A switcher will support either the **IBMDSwitcherSuperSourceBorder** interface or the **IBMDSwitcherSuperSourceBoxBorder** interface, not both.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputSuperSource	IID_IBMDSwitcherInputSuperSource	An IBMDSwitcherSuperSourceBorder object interface can be obtained with IBMDSwitcherInputSuperSource::QueryInterface .

Public Member Functions	
Method	Description
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.

Public Member Functions	
Method	Description
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.
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.

6.2.6.1 IBMDSwitcherSuperSourceBorder::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.

6.2.6.2 IBMDSwitcherSuperSourceBorder::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.

6.2.6.3 IBMDSwitcherSuperSourceBorder::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.

6.2.6.4 IBMDSwitcherSuperSourceBorder::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.

6.2.6.5 IBMDSwitcherSuperSourceBorder::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.

6.2.6.7 IBMDSwitcherSuperSourceBorder::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.

6.2.6.8 IBMDSwitcherSuperSourceBorder::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.

6.2.6.9 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.10 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.11 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.12 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.13 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.14 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.15 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.16 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.17 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.18 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.19 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.20 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.21 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.22 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.23 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.24 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.25 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.26 **IBMDSwitcherSuperSourceBorder::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.

6.2.6.27 IBMDSwitcherSuperSourceBorder::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.

6.2.7 IBMDSwitcherSuperSourceBorderCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSuperSourceBorder	IID_IBMDSwitcherSuperSourceBorder	An <code>IBMDSwitcherSuperSourceBorderCallback</code> object interface is installed with <code>IBMDSwitcherSuperSourceBorder::AddCallback</code> and removed with <code>IBMDSwitcherSuperSourceBorder::RemoveCallback</code> .

Public Member Functions

Method	Description
Notify	Called when an event occurs.

6.2.7.1 IBMDSwitcherSuperSourceBorderCallback::Notify method

The **Notify** method is called when **IBMDSwitcherSuperSourceBorder** 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(BMDSwitcherSuperSourceBorderEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherSuperSourceBorderEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8 IBMDSwitcherSuperSourceBoxBorder Interface

The **IBMDSwitcherSuperSourceBoxBorder** object interface is used for manipulating supersource border settings for a single supersource box. A switcher will support either the **IBMDSwitcherSuperSourceBorder** interface or the **IBMDSwitcherSuperSourceBoxBorder** interface, not both.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherSuperSourceBox	IID_IBMDSwitcherSuperSourceBox	An IBMDSwitcherSuperSourceBoxBorder object interface can be obtained with IBMDSwitcherSuperSourceBox::QueryInterface .

Public Member Functions	
Method	Description
GetBorderEnabled	Get the current border enabled flag.
SetBorderEnabled	Set the border enabled flag.
GetBorderWidthOutHorizontal	Get the current border outer horizontal width.
SetBorderWidthOutHorizontal	Set the border outer horizontal width.
GetBorderWidthOutVertical	Get the current border outer vertical width.
SetBorderWidthOutVertical	Set the border outer vertical width.
GetBorderWidthInLeft	Get the current border inner left width.
SetBorderWidthInLeft	Set the border inner left width.
GetBorderWidthInRight	Get the current border inner right width.
SetBorderWidthInRight	Set the border inner right width.

GetBorderWidthInTop	Get the current border inner top width.
SetBorderWidthInTop	Set the border inner top width.
GetBorderWidthInBottom	Get the current border inner bottom width.
SetBorderWidthInBottom	Set the border inner bottom width.
GetBorderHue	Get the current border hue.
SetBorderHue	Set the border hue.
GetBorderSaturation	Get the current border saturation.
SetBorderSaturation	Set the border saturation.
GetBorderLuma	Get the current border luminescence.
SetBorderLuma	Set the border luminescence.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

6.2.8.1 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.2 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.3 IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthOutHorizontal method

The `GetBorderWidthOutHorizontal` method returns the current border outer horizontal width.

Syntax

```
HRESULT GetBorderWidthOutHorizontal(double* widthOutHorizontal)
```

Parameters

Name	Direction	Description
widthOutHorizontal	out	The current border outer horizontal width.

Return Values

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

6.2.8.4 IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthOutHorizontal method

The `SetBorderWidthOutHorizontal` method sets the border outer horizontal width.

Syntax

```
HRESULT SetBorderWidthOutHorizontal(double widthOutHorizontal)
```

Parameters

Name	Direction	Description
widthOutHorizontal	in	The desired border outer horizontal width.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8.5 IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthOutVertical method

The `GetBorderWidthOutVertical` method returns the current border outer vertical width.

Syntax

```
HRESULT GetBorderWidthOutVertical(double* widthOutVertical)
```

Parameters

Name	Direction	Description
widthOutVertical	out	The current border outer vertical width.

Return Values

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

6.2.8.6 IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthOutVertical method

The `SetBorderWidthOutVertical` method sets the border outer vertical width.

Syntax

```
HRESULT SetBorderWidthOutVertical(double widthOutVertical)
```

Parameters

Name	Direction	Description
widthOutVertical	in	The desired border outer vertical width.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8.7 IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInLeft method

The `GetBorderWidthInLeft` method returns the current border inner left width.

Syntax

```
HRESULT GetBorderWidthInLeft(double* widthInLeft)
```

Parameters

Name	Direction	Description
widthInLeft	out	The current border inner left width.

Return Values

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

6.2.8.8 IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInLeft method

The `SetBorderWidthInLeft` method sets the border inner left width.

Syntax

```
HRESULT SetBorderWidthInLeft(double widthInLeft)
```

Parameters

Name	Direction	Description
widthInLeft	in	The desired border inner left width.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8.9 **IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInRight** method

The `GetBorderWidthInRight` method returns the current border inner right width.

Syntax

```
HRESULT GetBorderWidthInRight(double* widthInRight)
```

Parameters

Name	Direction	Description
widthInRight	out	The current border inner right width.

Return Values

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

6.2.8.10 **IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInRight** method

The `SetBorderWidthInRight` method sets the border inner right width.

Syntax

```
HRESULT SetBorderWidthInRight(double widthInRight)
```

Parameters

Name	Direction	Description
widthInRight	in	The desired border inner right width.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8.11 **IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInTop** method

The `GetBorderWidthInTop` method returns the current border inner top width.

Syntax

```
HRESULT GetBorderWidthInTop(double* widthInTop)
```

Parameters

Name	Direction	Description
widthInTop	out	The current border inner top width.

Return Values

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

6.2.8.12 **IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInTop** method

The **SetBorderWidthInTop** method sets the border inner top width.

Syntax

```
HRESULT SetBorderWidthInTop(double widthInTop)
```

Parameters

Name	Direction	Description
widthInTop	in	The desired border inner top width.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

6.2.8.13 **IBMDSwitcherSuperSourceBoxBorder::GetBorderWidthInBottom** method

The **GetBorderWidthInBottom** method returns the current border inner bottom width.

Syntax

```
HRESULT GetBorderWidthInBottom(double* widthInBottom)
```

Parameters

Name	Direction	Description
widthInBottom	out	The current border inner bottom width.

Return Values

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

6.2.8.14 **IBMDSwitcherSuperSourceBoxBorder::SetBorderWidthInBottom** method

The **SetBorderWidthInBottom** method sets the border inner bottom width.

Syntax

```
HRESULT SetBorderWidthInBottom(double widthInBottom)
```

Parameters

Name	Direction	Description
widthInBottom	in	The desired border inner bottom width.

Return Values

Value	Description
S_OK	Success.
E_POINTER	Failure.

6.2.8.15 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.16 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.17 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.18 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.19 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.20 **IBMDSwitcherSuperSourceBoxBorder::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.

6.2.8.21 IBMDSwitcherSuperSourceBoxBorder::AddCallback method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherSuperSourceBoxBorderCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherSuperSourceBoxBorderCallback object interface.

Return Values

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

6.2.8.22 IBMDSwitcherSuperSourceBoxBorder::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherSuperSourceBoxBorderCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherSuperSourceBoxBorderCallback object interface.

Return Values

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

6.2.9 IBMDSwitcherSuperSourceBoxBorderCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherSuperSourceBoxBorder</code>	<code>IID_IBMDSwitcherSuperSourceBoxBorder</code>	An <code>IBMDSwitcherSuperSourceBoxBorderCallback</code> object interface is installed with <code>IBMDSwitcherSuperSourceBoxBorder::AddCallback</code> and removed with <code>IBMDSwitcherSuperSourceBoxBorder::RemoveCallback</code> .

Public Member Functions	
Method	Description
<code>Notify</code>	Called when an event occurs.

6.2.9.1 IBMDSwitcherSuperSourceBoxBorderCallback::Notify method

The `Notify` method is called when `IBMDSwitcherSuperSourceBoxBorder` 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(BMDSwitcherSuperSourceBoxBorderEventType eventType)
```

Parameters

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

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

Section 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 Original Audio Mixing Data Types

7.1.1 Original Audio Mixer Event Type

BMDSwitcherAudioMixerEventType enumerates the possible event types for the **IBMDSwitcherAudioMixerCallback** object interface.

- **bmdSwitcherAudioMixerEventTypeProgramOutGainChanged**
The program out gain changed.
- **bmdSwitcherAudioMixerEventTypeProgramOutBalanceChanged**
The program out balance changed.

7.1.2 Original Audio Mixer Audio Input Identifier

BMDSwitcherAudioInputId is a signed 64 bit integer type used as a unique identifier for each audio input.

7.1.3 Original Audio Mixer Audio Output Identifier

BMDSwitcherAudioOutputId is an unsigned 16 bit integer type used as a unique identifier for each audio output.

7.1.4 Original Audio Mixer Audio Input Type

BMDSwitcherAudioInputType enumerates the possible input types for the **IBMDSwitcherAudioInput** object interface.

- **bmdSwitcherAudioInputTypeEmbeddedWithVideo**
The audio is embedded into a switcher input.
- **bmdSwitcherAudioInputTypeMediaPlayer**
The audio is from a media player.
- **bmdSwitcherAudioInputTypeAudioIn**
The audio is from an external audio-in.

7.1.5 Original Audio Mixer Mix Option

BMDSwitcherAudioMixOption enumerates the possible mix options for the **IBMDSwitcherAudioInput** object interface.

- **bmdSwitcherAudioMixOptionOff**
The audio is not to be mixed into anything.
- **bmdSwitcherAudioMixOptionOn**
The audio is always mixed into the output.
- **bmdSwitcherAudioMixOptionAudioFollowVideo**
The audio is mixed into the output when its associated video is on air.

7.1.6 Original Audio Mixer Audio Input Event Type

BMDSwitcherAudioInputEventType enumerates the possible event types for the **IBMDSwitcherAudioInputCallback** object interface.

- **bmdSwitcherAudioInputEventTypeMixOptionChanged**
The mix option changed.
- **bmdSwitcherAudioInputEventTypeGainChanged**
The gain changed.
- **bmdSwitcherAudioInputEventTypeBalanceChanged**
The balance changed.
- **bmdSwitcherAudioInputEventTypeCurrentExternalPortTypeChanged**
The audio input's external port type changed.
- **bmdSwitcherAudioInputEventTypesMixedInChanged**
The is-mixed-in changed.

7.1.7 Original Audio Mixer Monitor Output Event Type

BMDSwitcherAudioMonitorOutputEventType enumerates the possible event types for the **IBMDSwitcherAudioMonitorOutputCallback** object interface.

- **bmdSwitcherAudioMonitorOutputEventTypeMonitorEnableChanged**
The monitor enable flag changed.
- **bmdSwitcherAudioMonitorOutputEventTypeGainChanged**
The gain changed.
- **bmdSwitcherAudioMonitorOutputEventTypeMuteChanged**
The mute changed.
- **bmdSwitcherAudioMonitorOutputEventTypeSoloChanged**
The solo flag changed.
- **bmdSwitcherAudioMonitorOutputEventTypeSoloInputChanged**
The input that is soloed changed.
- **bmdSwitcherAudioMonitorOutputEventTypeDimChanged**
The dim flag changed.
- **bmdSwitcherAudioMonitorOutputEventTypeDimLevelChanged**
The dim level changed.

7.1.8 Original Audio Mixer Audio Headphone Output Event Type

BMDSwitcherAudioHeadphoneOutputEventType enumerates the possible event types for the **IBMDSwitcherAudioHeadphoneOutput** object interface.

- **bmdSwitcherAudioHeadphoneOutputEventTypeGainChanged**
The gain of the headphone output has changed.
- **bmdSwitcherAudioHeadphoneOutputEventTypeInputProgramOutGainChanged**
The gain of the program out input to the headphone output has changed.
- **bmdSwitcherAudioHeadphoneOutputEventTypeInputTalkbackGainChanged**
The gain of the talkback input to the headphone output has changed.
- **bmdSwitcherAudioHeadphoneOutputEventTypeInputSidetoneGainChanged**
The gain of the sidetone (microphone) input to the headphone output has changed.

7.2 Fairlight Audio Mixing Data Types

7.2.1 Fairlight Audio Mixer Event Type

BMDSwitcherFairlightAudioMixerEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioMixerCallback** object interface.

- **bmdSwitcherFairlightAudioMixerEventTypeMasterOutFaderGainChanged**
The master out fader gain changed.
- **bmdSwitcherFairlightAudioMixerEventTypeMasterOutFollowFadeToBlackChanged**
The master out follow fade to black changed.
- **bmdSwitcherFairlightAudioMixerEventTypeAudioFollowVideoCrossfadeTransitionChanged**
The audio follow video crossfade transition changed.
- **bmdSwitcherFairlightAudioMixerEventTypeMicTalkbackGainChanged**
The microphone gain changed.

7.2.2 Fairlight Audio Input Type

BMDSwitcherFairlightAudioInputType enumerates the possible input types for the **IBMDSwitcherFairlightAudioInput** object interface.

- **bmdSwitcherFairlightAudioInputTypeEmbeddedWithVideo**
The audio is from a camera input.
- **bmdSwitcherFairlightAudioInputTypeMediaPlayer**
The audio is from a media player.
- **bmdSwitcherFairlightAudioInputTypeAudioIn**
The audio is from an audio input.
- **bmdSwitcherFairlightAudioInputTypeMADI**
The audio is from a MADI input.

7.2.3 Fairlight Audio Input Configuration

BMDSwitcherFairlightAudioInputConfiguration enumerates the possible input types for the **IBMDSwitcherFairlightAudioInput** object interface.

- **bmdSwitcherFairlightAudioInputConfigurationMono**
The input configuration is mono.
- **bmdSwitcherFairlightAudioInputConfigurationStereo**
The input configuration is stereo.
- **bmdSwitcherFairlightAudioInputConfigurationDualMono**
The input configuration is dual mono.

7.2.4 Fairlight Audio Source Identifier

BMDSwitcherFairlightAudioSourceId is a signed 64-bit integer type used as a unique identifier for each Fairlight audio source.

7.2.5 Fairlight Audio Source Type

BMDSwitcherFairlightAudioSourceType enumerates the possible source types for the **IBMDSwitcherFairlightAudioSource** object interface.

- **bmdSwitcherFairlightAudioSourceTypeMono**
The source type is mono.
- **bmdSwitcherFairlightAudioSourceTypeStereo**
The source type is stereo.

7.2.6 Fairlight Audio Mix Option

BMDSwitcherFairlightAudioMixOption enumerates the possible mix options for the **IBMDSwitcherFairlightAudioSource** object interface.

- **bmdSwitcherFairlightAudioMixOptionOff**
The audio is not to be mixed into anything
- **bmdSwitcherFairlightAudioMixOptionOn**
The audio is always mixed into the output.
- **bmdSwitcherFairlightAudioMixOptionAudioFollowVideo**
The audio is mixed into the output when its associated video is on air.

7.2.7 Fairlight Audio Equalizer Band Shape

BMDSwitcherFairlightAudioEqualizerBandShape enumerates the possible band shapes for the **IBMDSwitcherFairlightAudioEqualizerBand** object interface.

- **bmdSwitcherFairlightAudioEqualizerBandShapeLowShelf**
The band shape is low shelf.
- **bmdSwitcherFairlightAudioEqualizerBandShapeLowPass**
The band shape is low pass.
- **bmdSwitcherFairlightAudioEqualizerBandShapeBandPass**
The band shape is band pass.
- **bmdSwitcherFairlightAudioEqualizerBandShapeNotch**
The band shape is notch.
- **bmdSwitcherFairlightAudioEqualizerBandShapeHighPass**
The band shape is high pass.
- **bmdSwitcherFairlightAudioEqualizerBandShapeHighShelf**
The band shape is high shelf.

7.2.8 Fairlight Audio Equalizer Band Frequency Range

BMDSwitcherFairlightAudioEqualizerBandFrequencyRange enumerates the possible frequency ranges for the **IBMDSwitcherFairlightAudioEqualizerBand** object interface.

- **bmdSwitcherFairlightAudioEqualizerBandFrequencyRangeLow**
The band frequency range is low.
- **bmdSwitcherFairlightAudioEqualizerBandFrequencyRangeMidLow**
The band frequency range is mid low.
- **bmdSwitcherFairlightAudioEqualizerBandFrequencyRangeMidHigh**
The band frequency range is mid high.
- **bmdSwitcherFairlightAudioEqualizerBandFrequencyRangeHigh**
The band frequency range is high.

7.2.9 Fairlight Audio Analog Input Level

BMDSwitcherFairlightAudioAnalogInputLevel enumerates the different input level settings for Fairlight analog audio inputs. This enumeration represents a bitset since an input may support multiple levels.

- **bmdSwitcherFairlightAudioAnalogInputLevelMicrophone**
Microphone input level.
- **bmdSwitcherFairlightAudioAnalogInputLevelConsumerLine**
Consumer line level (-10 dBV).
- **bmdSwitcherFairlightAudioAnalogInputLevelProLine**
Professional line level (+4 dBu).

7.2.10 Fairlight Audio Analog Input Mic Power Mode

BMDSwitcherFairlightAudioAnalogInputMicPowerMode enumerates the different microphone power mode settings for Fairlight analog audio inputs. This enumeration represents a bitset since an input may support multiple power modes.

- **bmdSwitcherFairlightAudioAnalogInputMicPowerModeNoPower**
No power supplied to microphone.
- **bmdSwitcherFairlightAudioAnalogInputMicPowerModePlugInPower**
Plug-in power is supplied to microphone

7.2.11 Fairlight Audio Input Event Type

BMDSwitcherFairlightAudioInputEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioInputCallback** object interface.

- **bmdSwitcherFairlightAudioInputEventTypeCurrentExternalPortTypeChanged**
The audio input's external port type changed.
- **bmdSwitcherFairlightAudioInputEventTypeConfigurationChanged**
The audio input's configuration changed.

7.2.12 Fairlight Audio Source Event Type

BMDSwitcherFairlightAudioSourceEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioSource** object interface.

- **bmdSwitcherFairlightAudioSourceEventTypesActiveChanged**
The is active property of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeMaxDelayFramesChanged**
The max delay frames of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeDelayFramesChanged**
The delay frames of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeInputGainChanged**
The input gain of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeStereoSimulationIntensityChanged**
The stereo simulation intensity of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypePanChanged**
The pan of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeFaderGainChanged**
The fader gain of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypeMixOptionChanged**
The mix option of the audio source has changed.
- **bmdSwitcherFairlightAudioSourceEventTypesMixedInChanged**
The mixed in of the audio source has changed.

7.2.13 Fairlight Audio Equalizer Event Type

BMDSwitcherFairlightAudioEqualizerEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioEqualizer** object interface.

- **bmdSwitcherFairlightAudioEqualizerEventTypeEnabledChanged**
The enable state of the equalizer has changed.
- **bmdSwitcherFairlightAudioEqualizerEventTypeGainChanged**
The gain of the equalizer has changed.

7.2.14 Fairlight Audio Equalizer Band Event Type

BMDSwitcherFairlightAudioEqualizerBandEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioEqualizerBand** object interface.

- **bmdSwitcherFairlightAudioEqualizerBandEventTypeEnabledChanged**
The enable of the equalizer band has changed.
- **bmdSwitcherFairlightAudioEqualizerBandEventTypeShapeChanged**
The shape of the equalizer band has changed.
- **bmdSwitcherFairlightAudioEqualizerBandEventTypeFrequencyRangeChanged**
The frequency range of the equalizer band has changed.
- **bmdSwitcherFairlightAudioEqualizerBandEventTypeFrequencyChanged**
The frequency of the equalizer band has changed.
- **bmdSwitcherFairlightAudioEqualizerBandEventTypeGainChanged**
The gain of the equalizer band has changed.
- **bmdSwitcherFairlightAudioEqualizerBandEventTypeQFactorChanged**
The Q factor of the equalizer band has changed.

7.2.15 Fairlight Audio Dynamics Event Type

BMDSwitcherFairlightAudioDynamicsProcessorEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioDynamicsProcessor** object interface.

- **bmdSwitcherFairlightAudioDynamicsProcessorEventTypeMakeupGainChanged**
The make up gain of the dynamics processor has changed.

7.2.16 Fairlight Audio Limiter Event Type

BMDSwitcherFairlightAudioLimiterEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioLimiter** object interface.

- **bmdSwitcherFairlightAudioLimiterEventTypeEnabledChanged**
The enable state of the limiter has changed.
- **bmdSwitcherFairlightAudioLimiterEventTypeThresholdChanged**
The threshold of the limiter has changed.
- **bmdSwitcherFairlightAudioLimiterEventTypeAttackChanged**
The attack of the limiter has changed.
- **bmdSwitcherFairlightAudioLimiterEventTypeHoldChanged**
The hold of the limiter has changed.
- **bmdSwitcherFairlightAudioLimiterEventTypeReleaseChanged**
The release of the limiter has changed.

7.2.17 Fairlight Audio Compressor Event Type

BMDSwitcherFairlightAudioCompressorEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioCompressor** object interface.

- **bmdSwitcherFairlightAudioCompressorEventTypeEnabledChanged**
The enable of the compressor has changed.
- **bmdSwitcherFairlightAudioCompressorEventTypeThresholdChanged**
The threshold of the compressor has changed.
- **bmdSwitcherFairlightAudioCompressorEventTypeRatioChanged**
The ratio of the compressor has changed.
- **bmdSwitcherFairlightAudioCompressorEventTypeAttackChanged**
The attack of the compressor has changed.
- **bmdSwitcherFairlightAudioCompressorEventTypeHoldChanged**
The hold of the compressor has changed.
- **bmdSwitcherFairlightAudioCompressorEventTypeReleaseChanged**
The release of the compressor has changed.

7.2.18 Fairlight Audio Expander Event Type

BMDSwitcherFairlightAudioExpanderEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioExpander** object interface.

- **bmdSwitcherFairlightAudioExpanderEventTypeEnabledChanged**
The enable state of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeGateModeChanged**
The gate mode of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeThresholdChanged**
The threshold of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeRangeChanged**
The range of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeRatioChanged**
The ratio of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeAttackChanged**
The attack of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeHoldChanged**
The hold of the expander has changed.
- **bmdSwitcherFairlightAudioExpanderEventTypeReleaseChanged**
The release of the expander has changed.

7.2.19 Fairlight Audio Headphone Output Event Type

BMDSwitcherFairlightAudioHeadphoneOutputEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioHeadphoneOutput** object interface.

- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeGainChanged**
The gain of the headphone output has changed.
- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeInput MasterOutGainChanged**
The gain of the master out input to the headphone output has changed.
- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeInput TalkbackGainChanged**
The gain of the talkback input to the headphone output has changed.
- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeInput SidetoneGainChanged**
The gain of the sidetone (microphone) input to the headphone output has changed.
- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeInputMasterOutMuteChanged**
The mute of the master out input to the headphone output has changed.
- **bmdSwitcherFairlightAudioHeadphoneOutputEventTypeInputTalkbackMuteChanged**
The mute of the talkback input to the headphone output has changed.

7.2.20 Fairlight Audio Solo Event Type

BMDSwitcherFairlightAudioSoloEventType enumerates the possible event types for **IBMDSwitcherFairlightAudioSoloCallback**.

- **bmdSwitcherFairlightAudioSoloEventTypeSoloChanged**
Solo state changed.
- **bmdSwitcherFairlightAudioSoloEventTypeSoloInputChanged**
Solo input changed.

7.2.21 Fairlight Analog Audio Input Event Type

BMDSwitcherFairlightAnalogAudioInputEventType enumerates the possible event types for **IBMDSwitcherFairlightAnalogAudioInputCallback**.

- **bmdSwitcherFairlightAnalogAudioInputEventTypeLevelChanged**
The analog input level setting changed.
- **bmdSwitcherFairlightAnalogAudioInputEventTypePowerModeChanged**
The analog input power mode setting changed.

7.2.22 Fairlight Audio Auxiliary Output Input Identifier

BMDSwitcherFairlightAudioAuxOutputInputId enumerates all the sources for the Fairlight audio auxiliary outputs.

- **BMDSwitcherFairlightAudioAuxOutputInputIdProgram**
The program mix is the current solo audio input, or the Program audio if no solo is selected.
- **BMDSwitcherFairlightAudioAuxOutputInputIdMicrophone**
The microphone audio is the audio from the talkback headset's microphone.
- **BMDSwitcherFairlightAudioAuxOutputInputIdProductionTalkback**
The production talkback consists of production talkback audio from the input cameras, talkback connector, and the headset microphone.
- **BMDSwitcherFairlightAudioAuxOutputInputIdEngineeringTalkback**
The engineering talkback consists of engineering talkback audio from the input cameras, talkback connector, and the headset microphone.

7.2.23 Fairlight Audio Auxiliary Output Identifier

BMDSwitcherFairlightAudioAuxOutputId enumerates all the Fairlight audio auxiliary outputs.

- **BMDSwitcherFairlightAudioAuxOutputIdControl**
Control audio output.
- **BMDSwitcherFairlightAudioAuxOutputIdStudio**
Studio audio output

7.2.24 Fairlight Audio Auxiliary Output Input Event Type

BMDSwitcherFairlightAudioAuxOutputInputEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioAuxOutputInputCallback** object interface.

- **bmdSwitcherFairlightAudioAuxOutputInputEventTypeGainChanged**
The gain level of the audio input has changed.
- **bmdSwitcherFairlightAudioAuxOutputInputEventTypeMuteChanged**
The mute state of the audio input has changed.

7.2.25 Fairlight Audio Auxiliary Output Event Type

BMDSwitcherFairlightAudioAuxOutputEventType enumerates the possible event types for the **IBMDSwitcherFairlightAudioAuxOutputCallback** object interface.

- **bmdSwitcherFairlightAudioAuxOutputEventTypeGainChanged**
The gain level of the audio output has changed.
- **bmdSwitcherFairlightAudioAuxOutputEventTypeMuteChanged**
The mute state of the audio output has changed.
- **bmdSwitcherFairlightAudioAuxOutputEventTypeDimOnChanged**
The dim state of the audio output has changed.
- **bmdSwitcherFairlightAudioAuxOutputEventTypeDimLevelChanged**
The dim level of the audio output has changed.

7.3 Talkback Data Types

7.3.1 Switcher Talkback Event Type

BMDSwitcherTalkbackEventType enumerates the possible event types for the **IBMDSwitcherTalkbackCallback** object interface.

- **bmdSwitcherTalkbackEventTypeMuteSDIChanged**
The mute state of the talkback input SDI channels has changed.
- **bmdSwitcherTalkbackEventTypeInputMuteSDIChanged**
The mute state of a talkback input SDI channel has changed.
- **bmdSwitcherTalkbackEventTypeCurrentInputSupportsMuteSDIChanged**
The input port has changed from/to an SDI port to/from a non-SDI port. As talkback is carried via SDI, the talkback mute setting has no effect when the input is on a non-SDI port.
- **bmdSwitcherTalkbackEventTypeMicrophoneMutedChanged**
The mute state of the talkback microphone has changed.

7.3.2 Switcher Talkback ID Type

BMDSwitcherTalkbackId enumerates the possible talkback ID types for the **IBMDSwitcherTalkback** object interface.

- **bmdSwitcherTalkbackIdProduction**
The talkback ID is a production talkback ID.
- **bmdSwitcherTalkbackIdEngineering**
The talkback ID is an engineering talkback ID.

7.4 Original Audio Mixing Interface Reference

7.4.1 IBMDSwitcherAudioMixer Interface

The **IBMDSwitcherAudioMixer** object interface is the root object for all original 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	IBMDSwitcher::QueryInterface can return an IBMDSwitcherAudioMixer 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.
GetAudioFollowVideoCrossfadeTransition	Get the current audio follow video crossfade transition state.
SetAudioFollowVideoCrossfadeTransition	Set the audio follow video crossfade transition 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.

7.4.1.1 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.

7.4.1.2 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.

7.4.1.3 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.

7.4.1.4 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.

7.4.1.5 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.

7.4.1.6 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.

7.4.1.7 IBMDSwitcherAudioMixer::GetAudioFollowVideoCrossfadeTransition

The `GetAudioFollowVideoCrossfadeTransition` method returns the current follow video with crossfade transition state. When enabled the audio will crossfade with the video.

Syntax

```
HRESULT GetAudioFollowVideoCrossfadeTransition(Boolean* transition)
```

Parameters

Name	Direction	Description
transition	out	The current follow video with crossfade transition state.

Return Values

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

7.4.1.8 **IBMDSwitcherAudioMixer::SetAudioFollowVideoCrossfadeTransition**

The **SetAudioFollowVideoCrossfadeTransition** method sets the current follow video with crossfade transition state. When enabled the audio will crossfade with the video.

Syntax

```
HRESULT SetAudioFollowVideoCrossfadeTransition(Boolean transition)
```

Parameters

Name	Direction	Description
transition	in	The desired follow video with crossfade transition state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.1.9 **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.

7.4.1.10 **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.

7.4.1.11 **IBMDSwitcherAudioMixer::ResetAllLevelNotificationPeaks** method

The **ResetAllLevelNotificationPeaks** method resets peak statistics to zero for all original audio mixer inputs and outputs.

Syntax

```
HRESULT ResetAllLevelNotificationPeaks(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.1.12 **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 IBMDSwitcherAudioMixerCallback object interface.

Return Values

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

7.4.113 **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 IBMDSwitcherAudioMixerCallback object interface.

Return Values

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

7.4.114 **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.

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 IBMDSwitcherAudioMixerCallback object interface is installed with IBMDSwitcherAudioMixer::AddCallback and removed with IBMDSwitcherAudioMixer::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
ProgramOutLevelNotification	Reports level statistics.

7.4.2.1 **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(IBMDSwitcherAudioMixerEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherAudioMixerEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.2.2 **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.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.3 **IBMDSwitcherAudioInputIterator** Interface

The **IBMDSwitcherAudioInputIterator** is used to enumerate the available inputs for the original 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	IBMDSwitcherAudioMixer::CreateIterator can return an IBMDSwitcherAudioInputIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherAudioInput object interface.
GetById	Returns a pointer to an IBMDSwitcherAudioInput object interface, given its BMDSwitcherAudioInputId .

7.4.3.1 IBMDSwitcherAudioInputIterator::Next method

The `Next` method returns the next available `IBMDSwitcherAudioInput` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherAudioInput* audioInput)
```

Parameters

Name	Direction	Description
audioInput	out	IBMDSwitcherAudioInput object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherAudioInput</code> objects available.
E_POINTER	The audioInput parameter is invalid.

7.4.3.2 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	<code>BMDSwitcherAudioInputId</code> identifier.
audioInput	out	IBMDSwitcherAudioInput 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.

7.4.4 IBMDSwitcherAudioInput Interface

The **IBMDSwitcherAudioInput** object interface is used for manipulating the settings of an original audio mixer audio input.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioInputIterator	IID_IBMDSwitcherAudioInputIterator	An IBMDSwitcherAudioInput object interface will be returned after a successful call to IBMDSwitcherAudioInputIterator::Next method.

Public Member Functions	
Method	Description
GetType	Get the audio input type.
GetCurrentExternalPortType	Get the current physical external port type of the audio input.
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	Returns the ID of this IBMDSwitcherAudioInput interface.
ResetLevelNotificationPeaks	Reset peak level statistics to zero.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.4.4.1 IBMDSwitcherAudioInput::GetType method

The **GetType** method returns the type of the original audio mixer audio input.

Syntax

```
HRESULT GetType(IBMDSwitcherAudioInputType* 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.

7.4.4.2 **IBMDSwitcherAudioInput::GetCurrentExternalPortType** method

The **GetCurrentExternalPortType** method gets the current physical external port type of the original audio mixer audio input. This may change if the physical input is switchable, generating the event **bmdSwitcherAudioInputEventTypeCurrentExternalPort TypeChanged**.

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.

7.4.4.3 **IBMDSwitcherAudioInput::GetMixOption** method

The **GetMixOption** method returns the mix option of the original audio mixer 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.

7.4.4.4 **IBMDSwitcherAudioInput::SetMixOption** method

The **SetMixOption** method sets the mix option of the original audio mixer 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.

7.4.4.5 **IBMDSwitcherAudioInput::GetGain** method

The **GetGain** method returns the gain currently applied to the original audio mixer 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.

7.4.4.6 IBMDSwitcherAudioInput::SetGain method

The **SetGain** method sets the gain to apply to the original audio mixer 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.

7.4.4.7 IBMDSwitcherAudioInput::GetBalance method

The **GetBalance** method returns the current balance for the original audio mixer audio input.

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.

7.4.4.8 IBMDSwitcherAudioInput::SetBalance method

The **SetBalance** method sets the balance to apply to the original audio mixer 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.

7.4.4.9 IBMDSwitcherAudioInput::IsMixedIn method

The **IsMixedIn** method indicates whether the original audio mixer 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.

7.4.4.10 **IBMDSwitcherAudioInput::GetAudioInputId** method

The **GetAudioInputId** method returns the **BMDSwitcherAudioInputId** of the original audio mixer audio input.

Syntax

```
HRESULT GetAudioInputId(BMDSwitcherAudioInputId* audioInputId)
```

Parameters

Name	Direction	Description
audioInputId	out	The BMDSwitcherAudioInputId of the audio input.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.4.11 **IBMDSwitcherAudioInput::ResetLevelNotificationPeaks** method

The **ResetLevelNotificationPeaks** method resets the switcher's input peak level statistics for the original audio mixer audio input to zero.

Syntax

```
HRESULT ResetLevelNotificationPeaks(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.4.12 **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 IBMDSwitcherAudioInputCallback object interface.

Return Values

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

7.4.4.13 **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 IBMDSwitcherAudioInputCallback object interface.

Return Values

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

7.4.5 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 IBMDSwitcherAudioInputCallback object interface is installed with IBMDSwitcherAudioInput::AddCallback and removed with IBMDSwitcherAudioInput::RemoveCallback

Public Member Functions

Method	Description
Notify	Called when an event occurs.
LevelNotification	Reports level statistics.

7.4.5.1 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(IBMDSwitcherAudioInputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherAudioInputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.5.2 IBMDSwitcherAudioInputCallback::LevelNotification method

The **LevelNotification** method is called periodically to report the current dB levels and the last known peak levels for the original audio mixer audio input. 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.

7.4.6 IBMDSwitcherAudioInputXLR Interface

The `IBMDSwitcherAudioInputXLR` object interface is used for manipulating the original audio mixer XLR input settings.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioInput</code>	<code>IID_IBMDSwitcherAudioInput</code>	<code>IBMDSwitcherAudioInput::QueryInterface</code> can return an <code>IBMDSwitcherAudioInputXLR</code> object interface.

Public Member Functions

Method	Description
<code>HasRCAToXLR</code>	Determine if the input has the capability of converting the input level from RCA to XLR.
<code>GetRCAToXLREnabled</code>	Query the RCA to XLR state of the XLR input.
<code>SetRCAToXLREnabled</code>	Set the RCA to XLR state of the XLR input.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.4.6.1 IBMDSwitcherAudioInputXLR::HasRCAToXLR method

The `HasRCAToXLR` method is used to check if the switcher input has the capability to convert the input level from RCA to XLR.

Syntax

```
HRESULT HasRCAToXLR(BOOLEAN* hasRcaToXlr)
```

Parameters

Name	Direction	Description
<code>hasRcaToXlr</code>	out	A Boolean value indicating whether the switcher input has the capability to convert the input level from RCA to XLR.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>hasRcaToXlr</code> parameter is not a valid pointer.

7.4.6.2 IBMDSwitcherAudioInputXLR::GetRCAToXLREnabled method

The **GetRCAToXLREnabled** method returns the RCA to XLR enabled state of the original audio mixer audio input.

Syntax

```
HRESULT GetRCAToXLREnabled(boolean* rcaToXlrEnabled)
```

Parameters

Name	Direction	Description
rcaToXlrEnabled	out	The RCA to XLR state of the audio input

Return Values

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

7.4.6.3 IBMDSwitcherAudioInputXLR::SetRCAToXLREnabled method

The **SetRCAToXLREnabled** method sets the RCA to XLR enabled state of the original audio mixer audio input.

Syntax

```
HRESULT SetRCAToXLREnabled(boolean rcaToXlrEnabled)
```

Parameters

Name	Direction	Description
rcaToXlrEnabled	in	The RCA to XLR state of the audio input

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.6.4 IBMDSwitcherAudioInputXLR::AddCallback method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherAudioInputXLRCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioInputXLRCallback object interface.

Return Values

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

7.4.6.5 IBMDSwitcherAudioInputXLR::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherAudioInputXLRCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioInputXLRCallback object interface.

Return Values

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

7.4.7 IBMDSwitcherAudioInputXLRCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioInputXLR	IID_ IBMDSwitcherAudioInputXLR	An IBMDSwitcherAudioInputXLRCallback object interface is installed with IBMDSwitcherAudioInputXLR::AddCallback and removed with IBMDSwitcherAudioInputXLR::RemoveCallback

Public Member Functions

Method	Description
Notify	Called when an event occurs.

7.4.7.1 IBMDSwitcherAudioInputXLRCallback::Notify method

The **Notify** method is called when **IBMDSwitcherAudioInputXLR** 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(BMDSwitcherAudioInputXLREventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherAudioInputXLREventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.8 IBMDSwitcherAudioMonitorOutputIterator Interface

The `IBMDSwitcherAudioMonitorOutputIterator` is used to enumerate the available monitor outputs for the original 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
<code>IBMDSwitcherAudioMixer</code>	<code>IID_IBMDSwitcherAudioMixer</code>	<code>IBMDSwitcherAudioMixer::CreateIterator</code> can return an <code>IBMDSwitcherAudioMonitorOutputIterator</code> object interface.

Public Member Functions	
Method	Description
<code>Next</code>	Returns a pointer to the next <code>IBMDSwitcherAudioMonitorOutputIterator</code> object interface.

7.4.8.1 IBMDSwitcherAudioMonitorOutputIterator::Next method

The `Next` method returns the next available `IBMDSwitcherAudioMonitorOutput` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherAudioMonitorOutput* audioMonitorOutput)
```

Parameters

Name	Direction	Description
<code>audioMonitorOutput</code>	out	<code>IBMDSwitcherAudioMonitorOutput</code> object interface.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>S_FALSE</code>	No more <code>IBMDSwitcherAudioMonitorOutput</code> objects available.
<code>E_POINTER</code>	The <code>audioMonitorOutput</code> parameter is invalid.

7.4.9 IBMDSwitcherAudioMonitorOutput Interface

The `IBMDSwitcherAudioMonitorOutput` object interface is used for manipulating parameters specific to audio monitor outputs for the original audio mixer.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioMonitorOutputIterator</code>	<code>IID_IBMDSwitcherAudioMonitorOutput</code>	An <code>IBMDSwitcherAudioMonitorOutput</code> interface can be obtained with <code>IBMDSwitcherAudioMonitorOutputIterator::Next</code> .

Public Member Functions	
Method	Description
<code>GetMonitorEnable</code>	Get the current monitor-enable flag.
<code>SetMonitorEnable</code>	Set the monitor-enable flag.
<code>GetMute</code>	Get the current mute flag.
<code>SetMute</code>	Set the mute flag.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set the gain value.
<code>GetSolo</code>	Get the current solo flag.
<code>SetSolo</code>	Set the solo flag.
<code>GetSoloInput</code>	Get the current soloed input.
<code>SetSoloInput</code>	Set the soloed input.
<code>GetDim</code>	Get the current dim flag.
<code>SetDim</code>	Set the dim flag.
<code>GetDimLevel</code>	Get the current dim level.
<code>SetDimLevel</code>	Set the dim level.
<code>ResetLevelNotificationPeaks</code>	Reset peak level statistics to zero.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.4.9.1 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.

7.4.9.2 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.

7.4.9.3 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.

7.4.9.4 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.

7.4.9.5 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.

7.4.9.6 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.

7.4.9.7 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.

7.4.9.8 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.

7.4.9.9 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.

7.4.9.10 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.

7.4.9.11 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.

7.4.9.12 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.

7.4.9.13 IBMDSwitcherAudioMonitorOutput::GetDimLevel method

The `GetDimLevel` method returns the current dim level in dB for the original audio mixer.

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.

7.4.9.14 **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.

7.4.9.15 **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.

7.4.9.16 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 IBMDSwitcherAudioMonitorOutputCallback object interface.

Return Values

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

7.4.9.17 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 IBMDSwitcherAudioMonitorOutputCallback object interface.

Return Values

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

7.4.10 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 IBMDSwitcherAudioMonitorOutputCallback object interface is installed with IBMDSwitcherAudioMonitorOutput::AddCallback and removed with IBMDSwitcherAudioMonitorOutput::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
LevelNotification	Reports level statistics.

7.4.10.1 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	BMDSwitcherAudioMonitorOutputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.10.2 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.

7.4.11 IBMDSwitcherAudioHeadphoneOutputIterator Interface

The `IBMDSwitcherAudioHeadphoneOutputIterator` is used to enumerate the available headphone outputs for the original audio mixer.

A reference to an `IBMDSwitcherAudioHeadphoneOutputIterator` object interface may be obtained from an `IBMDSwitcherAudioMixer` object interface using the `CreateIterator` method.

Pass `IID_IBMDSwitcherAudioHeadphoneOutputIterator` for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioMixer</code>	<code>IID_IBMDSwitcherAudioMixer</code>	<code>IBMDSwitcherAudioMixer::CreateIterator</code> returns an <code>IBMDSwitcherAudioHeadphoneOutputIterator</code> interface when the <code>IID_IBMDSwitcherAudioHeadphoneOutputIterator</code> IID is specified.

Public Member Functions	
Method	Description
<code>Next</code>	Returns a pointer to the next <code>IBMDSwitcherAudioHeadphoneOutput</code> object interface.

7.4.11.1 IBMDSwitcherAudioHeadphoneOutputIterator::Next method

The `Next` method returns the next available `IBMDSwitcherAudioHeadphoneOutput` object interface.

The `IBMDSwitcherAudioHeadphoneOutput` object interface must be released by the caller when no longer required.

Syntax

```
HRESULT Next(IBMDSwitcherAudioHeadphoneOutput* audioHeadphoneOutput)
```

Parameters

Name	Direction	Description
<code>audioHeadphoneOutput</code>	out	<code>IBMDSwitcherAudioHeadphoneOutput</code> object interface.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>S_FALSE</code>	No (more) headphone outputs are available.
<code>E_POINTER</code>	The <code>audioHeadphoneOutput</code> parameter is not a valid pointer.

7.4.12 IBMDSwitcherAudioHeadphoneOutput Interface

The `IBMDSwitcherAudioHeadphoneOutput` object interface is used for manipulating parameters specific to the original audio mixer headphone outputs.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioHeadphoneOutputIterator</code>	<code>IID_IBMDSwitcherAudioHeadphoneOutputIterator</code>	An <code>IBMDSwitcherAudioHeadphoneOutput</code> interface can be obtained with <code>IBMDSwitcherAudioHeadphoneOutputIterator::Next</code> .

Public Member Functions

Method	Description
<code>GetGain</code>	Get the current headphone output gain value.
<code>SetGain</code>	Set the headphone output gain value.
<code>GetInputProgramOutGain</code>	Get the current headphone program out input gain value.
<code>SetInputProgramOutGain</code>	Set the headphone program out input gain value.
<code>GetInputTalkbackGain</code>	Get the current headphone talkback input gain value.
<code>SetInputTalkbackGain</code>	Set the headphone talkback input gain value.
<code>GetInputSidetoneGain</code>	Get the current headphone sidetone (microphone) input gain value.
<code>SetInputSidetoneGain</code>	Set the headphone sidetone (microphone) input gain value.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.4.12.1 IBMDSwitcherAudioHeadphoneOutput::GetGain method

The `GetGain` method returns the gain currently applied to the headphone output.

Syntax

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

Parameters

Name	Direction	Description
<code>gain</code>	out	The current gain value.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The gain parameter is not a valid pointer.

7.4.12.2 IBMDSwitcherAudioHeadphoneOutput::SetGain method

The **SetGain** method sets the gain to apply to the headphone 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.

7.4.12.3 IBMDSwitcherAudioHeadphoneOutput::GetInputProgramOutGain method

The **GetInputProgramOutGain** method returns the gain currently applied to the program out input of the headphone output.

Syntax

```
HRESULT GetInputProgramOutGain(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 not a valid pointer.

7.4.12.4 **IBMDSwitcherAudioHeadphoneOutput::SetInputProgramOutGain** method

The **SetInputProgramOutGain** method sets the gain to apply to the program out input of the headphone output.

Syntax

```
HRESULT SetInputProgramOutGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The desired gain value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.12.5 **IBMDSwitcherAudioHeadphoneOutput::GetInputTalkbackGain** method

The **GetInputTalkbackGain** method returns the gain currently applied to the talkback input of the headphone output.

Syntax

```
HRESULT GetInputTalkbackGain(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 not a valid pointer.

7.4.12.6 **IBMDSwitcherAudioHeadphoneOutput::SetInputTalkbackGain** method

The **SetInputTalkbackGain** method sets the gain to apply to the talkback input of the headphone output.

Syntax

```
HRESULT SetInputTalkbackGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The desired gain value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.12.7 **IBMDSwitcherAudioHeadphoneOutput::GetInputSidetoneGain** method

The **GetInputSidetoneGain** method returns the gain currently applied to the sidetone (microphone) input of the headphone output.

Syntax

```
HRESULT GetInputSidetoneGain(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 not a valid pointer.

7.4.12.8 **IBMDSwitcherAudioHeadphoneOutput::SetInputSidetoneGain** method

The **SetInputSidetoneGain** method sets the gain to apply to the sidetone (microphone) input of the headphone output.

Syntax

```
HRESULT SetInputSidetoneGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The desired gain value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.4.12.9 **IBMDSwitcherAudioHeadphoneOutput::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherAudioHeadphoneOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioHeadphoneOutputCallback object interface.

Return Values

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

7.4.12.10 **IBMDSwitcherAudioHeadphoneOutput::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherAudioHeadphoneOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioHeadphoneOutputCallback object interface.

Return Values

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

7.4.13 IBMDSwitcherAudioHeadphoneOutputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioHeadphoneOutput	IID_IBMDSwitcherAudioHeadphoneOutput	An IBMDSwitcherAudioHeadphoneOutputCallback object interface is installed with IBMDSwitcherAudioHeadphoneOutput::AddCallback and removed with IBMDSwitcherAudioHeadphoneOutput::RemoveCallback .

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

7.4.13.1 IBMDSwitcherAudioHeadphoneOutputCallback::Notify method

The **Notify** method is called when **IBMDSwitcherAudioHeadphoneOutput** events occur, such as gain setting 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(BMDSwitcherAudioHeadphoneOutputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherAudioHeadphoneOutputEventType that describes the type of event that occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5 Fairlight Audio Mixing Interface Reference

7.5.1 IBMDSwitcherFairlightAudioMixer Interface

The **IBMDSwitcherFairlightAudioMixer** object interface is the root object for all Fairlight audio mixing control and feedback.

A reference to an **IBMDSwitcherFairlightAudioMixer** object interface may be obtained from an **IBMDSwitcher** object interface using the **QueryInterface** method.

Pass **IID_IBMDSwitcherFairlightAudioMixer** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::QueryInterface can return an IBMDSwitcherFairlightAudioMixer object interface.

Public Member Functions	
Method	Description
GetMasterOutEffect	Get a master out effect object interface.
GetMasterOutFaderGain	Get the current master out fader gain value.
SetMasterOutFaderGain	Set the master out fader gain value.
DoesSupportMicTalkbackGain	Determine if gain is supported by the talkback microphone.
GetMicTalkbackGain	Get the current talkback microphone gain value.
SetMicTalkbackGain	Set the talkback microphone gain value.
GetMasterOutFollowFadeToBlack	Get the current master out follow fade to black state.
SetMasterOutFollowFadeToBlack	Set the master out follow fade to black state.
GetAudioFollowVideoCrossfadeTransition	Get the current audio follow video crossfade transition state.
SetAudioFollowVideoCrossfadeTransition	Set the audio follow video crossfade transition state.
SetAllLevelNotificationsEnabled	Opt-in to level notifications.
ResetMasterOutPeakLevels	Reset master out peak levels.
ResetAllPeakLevels	Reset all switcher peak level statistics.
CreateIterator	Create an iterator.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.1.1 **IBMDSwitcherFairlightAudioMixer::GetMasterOutEffect** method

The **GetMasterOutEffect** method returns the master out effect object interface for the specified interface ID, such as **IBMDSwitcherFairlightAudioEqualizer** and **IBMDSwitcherFairlightAudioDynamicsProcessor**.

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

Syntax

```
HRESULT GetMasterOutEffect(REFIID iid, LPVOID* ppv)
```

Parameters

Name	Direction	Description
iid	in	Interface ID for the desired interface.
ppv	out	Pointer to return interface object.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The ppv parameter is invalid.
E_NOINTERFACE	Interface was not found.

7.5.1.2 **IBMDSwitcherFairlightAudioMixer::GetMasterOutFaderGain** method

The **GetMasterOutFaderGain** method returns the current gain value.

Syntax

```
HRESULT GetMasterOutFaderGain(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.

7.5.1.3 **IBMDSwitcherFairlightAudioMixer::SetMasterOutFaderGain** method

The **SetMasterOutFaderGain** method sets the gain to apply to the master out.

Syntax

```
HRESULT SetMasterOutFaderGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The desired gain value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.1.4 **IBMDSwitcherFairlightAudioMixer::DoesSupportMicTalkbackGain** method

The **DoesSupportMicTalkbackGain** method is used to determine whether gain control is supported by the talkback microphone audio input.

Syntax

```
HRESULT DoesSupportMicTalkbackGain(Boolean* supportsMicTalkbackGain)
```

Parameters

Name	Direction	Description
supportsMicTalkbackGain	out	Boolean value describing whether gain control is supported by the talkback microphone.

Return Values

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

7.5.1.5 **IBMDSwitcherFairlightAudioMixer::GetMicTalkbackGain** method

The **GetMicTalkbackGain** method returns the current gain value applied to the talkback microphone.

Syntax

```
HRESULT GetMicTalkbackGain(double* gain)
```

Parameters

Name	Direction	Description
gain	out	The gain currently applied to the talkback microphone.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The gain parameter is invalid.
E_NOTIMPL	Talkback microphone gain is not supported.

7.5.1.6 IBMDSwitcherFairlightAudioMixer::SetMicTalkbackGain method

The `SetMicTalkbackGain` method sets the gain value to apply to the talkback microphone.

Syntax

```
HRESULT SetMicTalkbackGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The gain to apply to the talkback microphone.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_NOTIMPL	Talkback microphone gain is not supported.

7.5.1.7 IBMDSwitcherFairlightAudioMixer::GetMasterOutFollowFadeToBlack method

The `GetMasterOutFollowFadeToBlack` method returns the current follow fade to black state. When enabled the master out audio will fade in unity with a fade to black transition.

Syntax

```
HRESULT GetMasterOutFollowFadeToBlack(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 follow parameter is invalid.

7.5.1.8 **IBMDSwitcherFairlightAudioMixer::SetMasterOutFollowFadeToBlack** method

The **SetMasterOutFollowFadeToBlack** method sets the current follow fade to black state. When enabled the master out audio will fade in unity with a fade to black transition.

Syntax

```
HRESULT SetMasterOutFollowFadeToBlack(boolean follow)
```

Parameters

Name	Direction	Description
follow	in	The desired follow fade to black state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.1.9 **IBMDSwitcherFairlightAudioMixer::GetAudioFollowVideoCrossfadeTransition** method

The **GetAudioFollowVideoCrossfadeTransition** method returns the current follow video with crossfade transition state. When enabled the audio will crossfade with the video.

Syntax

```
HRESULT GetAudioFollowVideoCrossfadeTransition(boolean* transition)
```

Parameters

Name	Direction	Description
transition	out	The current follow video with crossfade transition state.

Return Values

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

7.5.1.10 **IBMDSwitcherFairlightAudioMixer::SetAudioFollowVideoCrossfadeTransition** method

The **SetAudioFollowVideoCrossfadeTransition** method sets the current follow video with crossfade transition state. When enabled the audio will crossfade with the video.

Syntax

```
HRESULT SetAudioFollowVideoCrossfadeTransition(boolean transition)
```

Parameters

Name	Direction	Description
transition	in	The desired follow video with crossfade transition state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.1.11 **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled** method

The **SetAllLevelNotificationsEnabled** method enables level statistics for the Fairlight mixer inputs and outputs. Receiving level notifications are an opt-in subscription, affecting the callbacks

IBMDSwitcherFairlightAudioMixerCallback::MasterOutLevelNotification,
IBMDSwitcherFairlightAudioSourceCallback::OutputLevelNotification,
IBMDSwitcherFairlightAudioDynamicsProcessorCallback::InputLevelNotification,
IBMDSwitcherFairlightAudioDynamicsProcessorCallback::OutputLevelNotification,
IBMDSwitcherFairlightAudioLimiterCallback::GainReductionLevelNotification,
IBMDSwitcherFairlightAudioCompressorCallback::GainReductionLevelNotification and
IBMDSwitcherFairlightAudioExpanderCallback::GainReductionLevelNotification

Syntax

```
HRESULT SetAllLevelNotificationsEnabled(boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	Whether to enable notifications.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.1.12 **IBMDSwitcherFairlightAudioMixer::ResetMasterOutPeakLevels** method

The **ResetMasterOutPeakLevels** method resets the switcher's master out peak level statistics.

Syntax

HRESULT ResetMasterOutPeakLevels(void)

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.1.13 **IBMDSwitcherFairlightAudioMixer::ResetAllPeakLevels** method

The **ResetAllPeakLevels** method resets peak level statistics for all Fairlight audio mixer inputs and outputs.

Syntax

HRESULT ResetAllPeakLevels(void)

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.114 **IBMDSwitcherFairlightAudioMixer::CreateIterator method**

The **CreateIterator** method creates an iterator object interface for the specified interface ID, such as **IBMDSwitcherFairlightAudioInputIterator** and **IBMDSwitcherFairlightAudioHeadphoneOutputIterator**.

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.
E_NOINTERFACE	Interface was not found.

7.5.115 **IBMDSwitcherFairlightAudioMixer::AddCallback method**

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioMixerCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioMixerCallback object interface.

Return Values

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

7.5.1.16 IBMDSwitcherFairlightAudioMixer::RemoveCallback method

The `RemoveCallback` method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioMixerCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherFairlightAudioMixerCallback</code> object interface.

Return Values

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

7.5.2 IBMDSwitcherFairlightAudioMixerCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioMixer</code>	<code>IID_IBMDSwitcherFairlightAudioMixer</code>	An <code>IBMDSwitcherFairlightAudioMixerCallback</code> object interface is installed with <code>IBMDSwitcherFairlightAudioMixer::AddCallback</code> and removed with <code>IBMDSwitcherFairlightAudioMixer::RemoveCallback</code> .

Public Member Functions	
Method	Description
<code>Notify</code>	Called when an event occurs.
<code>MasterOutLevelNotification</code>	Reports level statistics.

7.5.2.1 **IBMDSwitcherFairlightAudioMixerCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherFairlightAudioMixer** 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(BMDSwitcherFairlightAudioMixerEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherFairlightAudioMixerEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.2.2 **IBMDSwitcherFairlightAudioMixerCallback::MasterOutLevelNotification** method

The **MasterOutLevelNotification** method is called periodically to report the current dB levels and the last known peak levels. These peak levels can be reset using **IBMDSwitcherFairlightAudioMixer::ResetMasterOutPeakLevels**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 MasterOutLevelNotification(uint32_t numLevels, const double* levels, uint32_t numPeakLevels, const double* peakLevels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of levels of the master out.
levels	in	The current dB levels of the master out.
numPeakLevels	in	The number of peak levels of the master out.
peakLevels	in	The highest encountered peak dB level of the master out since the last reset.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.3 IBMDSwitcherFairlightAudioInputIterator Interface

The **IBMDSwitcherFairlightAudioInputIterator** is used to enumerate the available audio inputs for the Fairlight audio mixer.

A reference to an **IBMDSwitcherFairlightAudioInputIterator** object interface may be obtained from an **IBMDSwitcherFairlightAudioMixer** object interface using the **CreateIterator** method.

Pass **IID_IBMDSwitcherFairlightAudioInputIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioMixer	IID_IBMDSwitcherFairlightAudioMixer	IBMDSwitcherFairlightAudioMixer::CreateIterator can return an IBMDSwitcherFairlightAudioInputIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherFairlightAudioInput object interface.
GetById	Returns a pointer to an IBMDSwitcherFairlightAudioInput object interface, given its BMDSwitcherAudioInputId .

7.5.3.1 IBMDSwitcherFairlightAudioInputIterator::Next method

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

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioInput* audioInput)
```

Parameters

Name	Direction	Description
audioInput	out	IBMDSwitcherFairlightAudioInput object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherFairlightAudioInput objects available.
E_POINTER	The audioInput parameter is invalid.

7.5.3.2 IBMDSwitcherFairlightAudioInputIterator::GetById method

The `GetById` method returns a pointer to an `IBMDSwitcherFairlightAudioInput` object interface, given its `IBMDSwitcherAudioInputId`.

Syntax

```
HRESULT GetById(IBMDSwitcherAudioInputId audioInputId,  
               IBMDSwitcherFairlightAudioInput* audioInput)
```

Parameters

Name	Direction	Description
audioInputId	in	<code>IBMDSwitcherAudioInputId</code> identifier.
audioInput	out	<code>IBMDSwitcherFairlightAudioInput</code> object interface.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_INVALIDARG</code>	The <code>audioInputId</code> is not a valid identifier.
<code>E_POINTER</code>	The <code>audioInput</code> parameter is invalid.

7.5.4 IBMDSwitcherFairlightAudioInput Interface

The `IBMDSwitcherFairlightAudioInput` object interface is used for managing a Fairlight audio input.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioInputIterator</code>	<code>IID_IBMDSwitcherFairlightAudioInputIterator</code>	An <code>IBMDSwitcherFairlightAudioInput</code> object interface will be returned after a successful call to <code>IBMDSwitcherFairlightAudioInputIterator::Next</code> method.

Public Member Functions	
Method	Description
<code>GetType</code>	Get the audio input type.
<code>GetCurrentExternalPortType</code>	Get the current physical external port type of the Fairlight audio input.
<code>GetSupportedConfigurations</code>	Get the available input configurations.
<code>GetConfiguration</code>	Get the current input configuration.
<code>SetConfiguration</code>	Set the input configuration.
<code>GetId</code>	Returns the ID of this <code>IBMDSwitcherFairlightAudioInput</code> interface.
<code>CreateIterator</code>	Create an iterator.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.4.1 **IBMDSwitcherFairlightAudioInput::GetType** method

The **GetType** method returns the type of the Fairlight audio input.

Syntax

```
HRESULT GetType(BMDSwitcherFairlightAudioInputType* type)
```

Parameters

Name	Direction	Description
type	out	The Fairlight audio input type.

Return Values

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

7.5.4.2 **IBMDSwitcherFairlightAudioInput::GetCurrentExternalPortType** method

The **GetCurrentExternalPortType** method gets the current physical external port type of the Fairlight audio input. This may change if the physical input is switchable, generating the event **bmdSwitcherFairlightAudioInputEventTypeCurrentExternalPortTypeChanged**.

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.

7.5.4.3 **IBMDSwitcherFairlightAudioInput::GetSupportedConfigurations** method

The **GetSupportedConfigurations** method returns the supported input configurations of the Fairlight audio input.

Syntax

```
HRESULT GetSupportedConfigurations(IBMDSwitcherFairlightAudioInputConfiguration* supportedConfigurations)
```

Parameters

Name	Direction	Description
supportedConfigurations	out	The supported Fairlight audio input configurations.

Return Values

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

7.5.4.4 **IBMDSwitcherFairlightAudioInput::GetConfiguration** method

The **GetConfiguration** method returns the current Fairlight audio input configuration.

Syntax

```
HRESULT GetConfiguration(IBMDSwitcherFairlightAudioInputConfiguration* configuration)
```

Parameters

Name	Direction	Description
configuration	out	The current Fairlight audio input configuration.

Return Values

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

7.5.4.5 **BMDSwitcherFairlightAudioInput::SetConfiguration** method

The **SetConfiguration** method sets the Fairlight audio input configuration.

Syntax

```
HRESULT SetConfiguration(BMDSwitcherFairlightAudioInputConfiguration configuration)
```

Parameters

Name	Direction	Description
configuration	in	The Fairlight audio input configuration

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The configuration is not a valid identifier.

7.5.4.6 **BMDSwitcherFairlightAudioInput::GetId** method

The **GetId** method returns the audio input's ID, used to uniquely identify an audio input within the Switcher.

Syntax

```
HRESULT GetId(BMDSwitcherAudioInputId* audioInputId)
```

Parameters

Name	Direction	Description
audioInputId	out	BMDSwitcherAudioInputId identifier for the current audio input.

Return Values

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

7.5.4.7 IBMDSwitcherFairlightAudioInput::CreateIterator method

The `CreateIterator` method creates an iterator object for the interface ID `IBMDSwitcherFairlightAudioSourceIterator`.

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_NOINTERFACE	Interface was not found.

7.5.4.8 IBMDSwitcherFairlightAudioInput::AddCallback method

The `AddCallback` method configures a callback to be called when events occur for an `IBMDSwitcherFairlightAudioInput` object. Pass an object implementing the `IBMDSwitcherFairlightAudioInputCallback` interface to receive callbacks.

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioInputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherFairlightAudioInputCallback</code> object interface.

Return Values

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

7.5.4.9 IBMDSwitcherFairlightAudioInput::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioInputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioInputCallback object interface.

Return Values

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

7.5.5 IBMDSwitcherFairlightAudioInputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioInput	IID_IBMDSwitcherFairlightAudioInput	An IBMDSwitcherFairlightAudioInputCallback object interface is installed with IBMDSwitcherFairlightAudioInput::AddCallback and removed with IBMDSwitcherFairlightAudioInput::RemoveCallback

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

7.5.5.1 **IBMDSwitcherFairlightAudioInputCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherFairlightAudioInput** 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(IBMDSwitcherFairlightAudioInputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioInputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.6 **IBMDSwitcherFairlightAudioSourceIterator** Interface

The **IBMDSwitcherFairlightAudioSourceIterator** is used to enumerate the available audio sources for a Fairlight audio mixer input.

A reference to an **IBMDSwitcherFairlightAudioSourceIterator** object interface may be obtained from an **IBMDSwitcherFairlightAudioInput** object interface using the **CreateIterator** method.

Pass **IID_IBMDSwitcherFairlightAudioSourceIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioInput	IID_IBMDSwitcherFairlightAudioInput	IBMDSwitcherFairlightAudioInput::CreateIterator can return an IBMDSwitcherFairlightAudioSourceIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherFairlightAudioSource object interface.
GetById	Returns a pointer to an IBMDSwitcherFairlightAudioSource object interface, given its IBMDSwitcherFairlightAudioSourceId .

7.5.6.1 IBMDSwitcherFairlightAudioSourceIterator::Next method

The `Next` method returns the next available `IBMDSwitcherFairlightAudioSource` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioSource* audioSource)
```

Parameters

Name	Direction	Description
audioSource	out	IBMDSwitcherFairlightAudioSource object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherFairlightAudioSource</code> objects available.
E_POINTER	The audioSource parameter is invalid.

7.5.6.2 IBMDSwitcherFairlightAudioSourceIterator::GetById method

The `GetById` method returns a pointer to an `IBMDSwitcherFairlightAudioSource` object interface, given its `BMDSwitcherFairlightAudioSourceId`.

Syntax

```
HRESULT GetById(BMDSwitcherFairlightAudioSourceId audioSourceId,  
IBMDSwitcherFairlightAudioSource* audioSource)
```

Parameters

Name	Direction	Description
audioSourceId	in	BMDSwitcherFairlightAudioSourceId identifier.
audioSource	out	IBMDSwitcherFairlightAudioSource object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioSourceId is not a valid identifier.
E_POINTER	The audioSource parameter is invalid.

7.5.7 IBMDSwitcherFairlightAudioSource Interface

The **IBMDSwitcherFairlightAudioSource** object interface is used for manipulating the settings of an audio source for the Fairlight audio mixer.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioSourceIteator	IID_IBMDSwitcherFairlightAudioSourceIteator	An IBMDSwitcherFairlightAudioSource object interface will be returned after a successful call to IBMDSwitcherFairlightAudioSourceIteator::Next method.

Public Member Functions	
Method	Description
IsActive	Get the current is-active state.
GetSourceType	Get the current source type.
GetMaxDelayFrames	Get the maximum delay frames.
GetDelayFrames	Get the current delay frames.
SetDelayFrames	Set the delay frames.
GetInputGain	Get the current input gain value.
SetInputGain	Set the input gain value.
HasStereoSimulation	Get the current has-stereo-simulation flag.
GetStereoSimulationIntensity	Get the current stereo-simulation-intensity percentage.
SetStereoSimulationIntensity	Set the stereo-simulation-intensity percentage.
GetEffect	Get the effect object interface.
GetPan	Get the current pan value.
SetPan	Set the pan value.
GetFaderGain	Get the current fader gain value.
SetFaderGain	Set the fader gain value.
GetSupportedMixOptions	Get the supported mix options.
GetMixOption	Get the current mix option.
SetMixOption	Set the mix option.
IsMixedIn	Get the is-mixed-in flag.
ResetOutputPeakLevels	Reset output peak level statistics.
GetId	Returns the ID of this IBMDSwitcherFairlightAudioSource interface.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.7.1 **IBMDSwitcherFairlightAudioSource::IsActive** method

The **IsActive** method indicates whether the Fairlight audio source is currently active.

Audio sources can become inactive when the configuration property of the **IBMDSwitcherFairlightAudioInput** is changed. When a source is not active, it can not be used to manipulate audio on the switcher.

Syntax

```
HRESULT IsActive(boolean* active)
```

Parameters

Name	Direction	Description
active	out	The current is-active state.

Return Values

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

7.5.7.2 **IBMDSwitcherFairlightAudioSource::GetSourceType** method

The **GetSourceType** method indicates the type of Fairlight audio source.

Syntax

```
HRESULT GetSourceType(BMDSwitcherFairlightAudioSourceType* type)
```

Parameters

Name	Direction	Description
type	out	The current Fairlight audio source type.

Return Values

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

7.5.7.3 **IBMDSwitcherFairlightAudioSource::GetMaxDelayFrames** method

The **GetMaxDelayFrames** method returns the maximum delay frames.

Syntax

```
HRESULT GetMaxDelayFrames(uint16_t* maxDelay)
```

Parameters

Name	Direction	Description
maxDelay	out	Maximum number of delay frames.

Return Values

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

7.5.7.4 **IBMDSwitcherFairlightAudioSource::GetDelayFrames** method

The **GetDelayFrames** method returns the current number of delay frames applied to the Fairlight audio source.

Syntax

```
HRESULT GetDelayFrames(uint16_t* delay)
```

Parameters

Name	Direction	Description
delay	out	The current number of delay frames applied to the Fairlight audio source.

Return Values

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

7.5.7.5 **IBMDSwitcherFairlightAudioSource::SetDelayFrames** method

The **SetDelayFrames** method sets the number of delay frames to apply to the Fairlight audio source.

Syntax

```
HRESULT SetDelayFrames(uint16_t delay)
```

Parameters

Name	Direction	Description
delay	in	The number of delay frames to apply to the Fairlight audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.6 **IBMDSwitcherFairlightAudioSource::GetInputGain** method

The **GetInputGain** method returns the current input gain applied to the Fairlight audio source.

Syntax

```
HRESULT GetInputGain(double* gain)
```

Parameters

Name	Direction	Description
gain	out	The gain currently applied to the Fairlight audio source.

Return Values

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

7.5.7.7 **IBMDSwitcherFairlightAudioSource::SetInputGain** method

The **SetInputGain** method sets the input gain of a Fairlight audio source.

Syntax

```
HRESULT SetInputGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The gain to apply to the audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.8 **IBMDSwitcherFairlightAudioSource::HasStereoSimulation** method

The **HasStereoSimulation** method indicates whether the Fairlight audio source has stereo simulation available.

Syntax

```
HRESULT HasStereoSimulation(boolean* hasStereoSimulation)
```

Parameters

Name	Direction	Description
hasStereoSimulation	out	The has-stereo-simulation flag.

Return Values

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

7.5.7.9 **IBMDSwitcherFairlightAudioSource::GetStereoSimulationIntensity** method

The **GetStereoSimulationIntensity** method returns the current intensity of the stereo simulation applied to the Fairlight audio source.

Syntax

```
HRESULT GetStereoSimulationIntensity(double* intensity)
```

Parameters

Name	Direction	Description
intensity	out	The current stereo-simulation-intensity percentage applied to the Fairlight audio source.

Return Values

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

7.5.7.10 **IBMDSwitcherFairlightAudioSource::SetStereoSimulationIntensity** method

The **SetStereoSimulationIntensity** method sets the intensity of the stereo simulation to apply to the Fairlight audio source.

Syntax

```
HRESULT SetStereoSimulationIntensity(double intensity)
```

Parameters

Name	Direction	Description
intensity	in	The desired stereo simulation intensity to apply to the Fairlight audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.11 **IBMDSwitcherFairlightAudioSource::GetEffect** method

The **GetEffect** method returns the effect object interface for the specified interface ID, such as **IBMDSwitcherFairlightAudioEqualizer** and **IBMDSwitcherFairlightAudioDynamicsProcessor**.

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

Syntax

```
HRESULT GetEffect(REFIID iid, LPVOID* ppv)
```

Parameters

Name	Direction	Description
iid	in	Interface ID for the desired interface.
ppv	out	Pointer to return interface object.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The ppv parameter is invalid.
E_NOINTERFACE	Interface was not found.

7.5.7.12 **IBMDSwitcherFairlightAudioSource::GetPan** method

The **GetPan** method returns the current pan value applied to the Fairlight audio source.

Syntax

```
HRESULT GetPan(double* pan)
```

Parameters

Name	Direction	Description
pan	out	The pan currently applied to the Fairlight audio source.

Return Values

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

7.5.7.13 **IBMDSwitcherFairlightAudioSource::SetPan** method

The **SetPan** method sets the pan value to apply to the Fairlight audio source.

Syntax

```
HRESULT SetPan(double pan)
```

Parameters

Name	Direction	Description
pan	in	The pan to apply to the Fairlight audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.14 **IBMDSwitcherFairlightAudioSource::GetFaderGain** method

The **GetFaderGain** method returns the current fader gain value applied to the Fairlight audio source.

Syntax

```
HRESULT GetFaderGain(double* gain)
```

Parameters

Name	Direction	Description
gain	out	The fader gain currently applied to the Fairlight audio source.

Return Values

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

7.5.7.15 **IBMDSwitcherFairlightAudioSource::SetFaderGain** method

The **SetFaderGain** method sets the fader gain value to apply to the Fairlight audio source.

Syntax

```
HRESULT SetFaderGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The fader gain value to apply to the Fairlight audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.16 **IBMDSwitcherFairlightAudioSource::GetSupportedMixOptions** method

The `GetSupportedMixOptions` method returns the supported mix options.

Syntax

```
HRESULT GetSupportedMixOptions (IBMDSwitcherFairlightAudioMixOption* mixOptions)
```

Parameters

Name	Direction	Description
mixOptions	out	The available mix options.

Return Values

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

7.5.7.17 **IBMDSwitcherFairlightAudioSource::GetMixOption** method

The `GetMixOption` method returns the current mix option.

Syntax

```
HRESULT GetMixOption(IBMDSwitcherFairlightAudioMixOption* mixOption)
```

Parameters

Name	Direction	Description
mixOption	out	The current mix option.

Return Values

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

7.5.7.18 **IBMDSwitcherFairlightAudioSource::SetMixOption** method

The `SetMixOption` method sets the mix option.

Syntax

```
HRESULT SetMixOption(IBMDSwitcherFairlightAudioMixOption mixOption)
```

Parameters

Name	Direction	Description
mixOption	in	The desired mix option.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.
E_INVALIDARG	The mixOption is not a valid identifier.

7.5.7.19 **IBMDSwitcherFairlightAudioSource::IsMixedIn** method

The **IsMixedIn** method indicates whether the Fairlight audio source 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.

7.5.7.20 **IBMDSwitcherFairlightAudioSource::ResetOutputPeakLevels** method

The **ResetOutputPeakLevels** method resets peak statistics for the Fairlight audio source.

Syntax

```
HRESULT ResetOutputPeakLevels(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure. This can happen if the source is no longer active.

7.5.7.21 **IBMDSwitcherFairlightAudioSource::GetId** method

The **GetId** method returns the **BMDSwitcherFairlightAudioSourceId** of the Fairlight audio source.

Syntax

```
HRESULT GetId(BMDSwitcherFairlightAudioSourceId* sourceId)
```

Parameters

Name	Direction	Description
sourceId	out	BMDSwitcherFairlightAudioSourceId identifier for the current Fairlight audio source.

Return Values

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

7.5.7.22 **IBMDSwitcherFairlightAudioSource::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioSourceCallback object interface.

Return Values

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

7.5.7.23 **IBMDSwitcherFairlightAudioSource::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioSourceCallback object interface.

Return Values

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

7.5.8 IBMDSwitcherFairlightAudioSourceCallback Interface

The **IBMDSwitcherFairlightAudioSourceCallback** object interface is a callback class containing methods that are called when an event occurs on an **IBMDSwitcherFairlightAudioSource** object.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioSource	IID_IBMDSwitcherFairlightAudioSource	An IBMDSwitcherFairlightAudioSourceCallback object interface is installed with IBMDSwitcherFairlightAudioSource::AddCallback and removed with IBMDSwitcherFairlightAudioSource::RemoveCallback .

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
OutputLevelNotification	Reports output level statistics.

7.5.8.1 IBMDSwitcherFairlightAudioSourceCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioSource** 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(IBMDSwitcherFairlightAudioSourceEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioSourceEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.8.2 IBMDSwitcherFairlightAudioSourceCallback::OutputLevelNotification method

The **OutputLevelNotification** method is called periodically to report the current dB output levels and the last known peak levels. These peak levels can be reset using **IBMDSwitcherFairlightAudioSource::ResetOutputPeakLevels**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 OutputLevelNotification(uint32_t numLevels, const double* levels, uint32_t numPeakLevels, const double* peakLevels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of output levels.
levels	in	The current output dB levels.
numPeakLevels	in	The number of output peak levels.
peakLevels	in	The highest encountered output peak level.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.9 IBMDSwitcherFairlightAudioEqualizer Interface

The `IBMDSwitcherFairlightAudioEqualizer` object interface is used for manipulating the Fairlight audio equalizer interface.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioMixer	IBMD_SwitcherFairlightAudioMixer	An <code>IBMDSwitcherFairlightAudioEqualizer</code> object interface will be returned after a successful call to <code>IBMDSwitcherFairlightAudioMixer::GetMasterOutEffect</code> method.
IBMDSwitcherFairlightAudioSource	IID_IBMDSwitcherFairlightAudioSource	An <code>IBMDSwitcherFairlightAudioEqualizer</code> object interface will be returned after a successful call to <code>IBMDSwitcherFairlightAudioSource::GetEffect</code> method.

Public Member Functions	
Method	Description
GetEnabled	Get the current equalizer-enabled flag.
SetEnabled	Set the equaliser-enabled flag.
GetGain	Get the current gain value.
SetGain	Set the gain value.
Reset	Reset the equalizer.
CreateIterator	Create an iterator.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.9.1 IBMDSwitcherFairlightAudioEqualizer::GetEnabled method

The `GetEnabled` method returns the current equalizer enabled flag.

Syntax

```
HRESULT GetEnabled(boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current equalizer enabled flag.

Return Values

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

7.5.9.2 **IBMDSwitcherFairlightAudioEqualizer::SetEnabled** method

The **SetEnabled** method sets the equalizer enabled flag.

Syntax

```
HRESULT SetEnabled(boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The desired equalizer enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.9.3 **IBMDSwitcherFairlightAudioEqualizer::GetGain** method

The **GetGain** method returns the current gain value applied to the Fairlight audio source by the equalizer.

Syntax

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

Parameters

Name	Direction	Description
gain	out	The gain currently applied to the Fairlight audio source by the equalizer.

Return Values

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

7.5.9.4 **IBMDSwitcherFairlightAudioEqualizer::SetGain** method

The **SetGain** method sets the gain value to apply to the Fairlight audio source.

Syntax

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

Parameters

Name	Direction	Description
gain	in	The gain to apply to the Fairlight audio source by the equalizer.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.9.5 **IBMDSwitcherFairlightAudioEqualizer::Reset** method

The **Reset** method resets the equalizer to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.9.6 **IBMDSwitcherFairlightAudioEqualizer::CreateIterator** method

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

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.
E_NOINTERFACE	Interface was not found.

7.5.9.7 **IBMDSwitcherFairlightAudioEqualizer::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioEqualizerCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioEqualizerCallback object interface.

Return Values

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

7.5.9.8 **IBMDSwitcherFairlightAudioEqualizer::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioEqualizerCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioEqualizerCallback object interface.

Return Values

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

7.5.10 IBMDSwitcherFairlightAudioEqualizerCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioEqualizer	IID_ IBMDSwitcherFairlightAudioEqualizer	An IBMDSwitcherFairlightAudioEqualizerCallback object interface is installed with IBMDSwitcherFairlightAudioEqualizer::AddCallback and removed with IBMDSwitcherFairlightAudioEqualizer::RemoveCallback

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

7.5.10.1 IBMDSwitcherFairlightAudioEqualizerCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioEqualizer** 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(IBMDSwitcherFairlightAudioEqualizerEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioEqualizerEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.11 IBMDSwitcherFairlightAudioEqualizerBandIterator Interface

The **IBMDSwitcherFairlightAudioEqualizerBandIterator** is used to enumerate the bands of a Fairlight audio equalizer.

A reference to an **IBMDSwitcherFairlightAudioEqualizerBandIterator** object interface may be obtained from an **IBMDSwitcherFairlightAudioEqualizer** object interface using the `CreateIterator` method.

Pass `IID_IBMDSwitcherFairlightAudioEqualizerBandIterator` for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioEqualizer	IID_IBMDSwitcherFairlightAudioEqualizer	IBMDSwitcherFairlightAudioEqualizer::CreateIterator can return an IBMDSwitcherFairlightAudioEqualizerBandIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherFairlightAudioEqualizerBand object interface.

7.5.11.1 IBMDSwitcherFairlightAudioEqualizerBandIterator::Next method

The `Next` method returns the next available **IBMDSwitcherFairlightAudioEqualizerBand** object interface.

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioEqualizerBand* audioEqualizerBand)
```

Parameters

Name	Direction	Description
audioEqualizerBand	out	IBMDSwitcherFairlightAudioEqualizerBand object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherFairlightAudioEqualizerBand objects available.
E_POINTER	The audioEqualizerBand parameter is invalid.

7.5.12 IBMDSwitcherFairlightAudioEqualizerBand Interface

The `IBMDSwitcherFairlightAudioEqualizerBand` object interface is used for manipulating Fairlight audio equalizer bands.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioEqualizerBandIterator</code>	<code>IID_IBMDSwitcherFairlightAudioEqualizerBandIterator</code>	An <code>IBMDSwitcherFairlightAudioEqualizerBand</code> interface can be obtained with <code>IBMDSwitcherFairlightAudioEqualizerBandIterator::Next</code> .

Public Member Functions	
Method	Description
<code>GetEnabled</code>	Get the current band-enabled flag.
<code>SetEnabled</code>	Set the band-enabled flag.
<code>GetSupportedShapes</code>	Get the supported shapes.
<code>GetShape</code>	Get the current band shape.
<code>SetShape</code>	Set the band shape.
<code>GetSupportedFrequencyRanges</code>	Get the supported frequency ranges.
<code>GetFrequencyRange</code>	Get the current frequency range.
<code>SetFrequencyRange</code>	Set the frequency range.
<code>GetFrequencyRangeMinMax</code>	Get the minimum and maximum frequencies for the specified frequency range.
<code>GetFrequency</code>	Get the current frequency value.
<code>SetFrequency</code>	Set the frequency value.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set the gain value.
<code>GetQFactor</code>	Get the current Q factor.
<code>SetQFactor</code>	Set the Q factor.
<code>Reset</code>	Reset the equalizer band.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.12.1 **IBMDSwitcherFairlightAudioEqualizerBand::GetEnabled** method

The **GetEnabled** method returns the current equalizer band enabled flag.

Syntax

```
HRESULT GetEnabled(BOOLEAN* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current band enabled flag.

Return Values

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

7.5.12.2 **IBMDSwitcherFairlightAudioEqualizerBand::SetEnabled** method

The **SetEnabled** method sets the equalizer band enabled flag.

Syntax

```
HRESULT SetEnabled(BOOLEAN enabled)
```

Parameters

Name	Direction	Description
enabled	in	The desired band enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.12.3 **IBMDSwitcherFairlightAudioEqualizerBand::GetSupportedShapes** method

The **GetSupportedShapes** method returns the supported equalizer band shapes.

Syntax

```
HRESULT GetSupportedShapes(BMDSwitcherFairlightAudioEqualizerBandShape* shapes)
```

Parameters

Name	Direction	Description
shapes	out	The available shapes.

Return Values

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

7.5.12.4 **IBMDSwitcherFairlightAudioEqualizerBand::GetShape** method

The **GetShape** method returns the current equalizer band shape.

Syntax

```
HRESULT GetShape(BMDSwitcherFairlightAudioEqualizerBandShape* shape)
```

Parameters

Name	Direction	Description
shape	out	The current shape.

Return Values

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

7.5.12.5 **IBMDSwitcherFairlightAudioEqualizerBand::SetShape** method

The **SetShape** method sets the equalizer band shape.

Syntax

```
HRESULT SetShape(BMDSwitcherFairlightAudioEqualizerBandShape shape)
```

Parameters

Name	Direction	Description
shape	in	The desired shape.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The shape is not a valid identifier.

7.5.12.6 **IBMDSwitcherFairlightAudioEqualizerBand::GetSupportedFrequencyRanges** method

The **GetSupportedFrequencyRanges** method returns the available frequency ranges for the equalizer band.

Syntax

```
HRESULT GetSupportedFrequencyRanges  
(BMDSwitcherFairlightAudioEqualizerBand FrequencyRange* ranges)
```

Parameters

Name	Direction	Description
ranges	out	The available frequency ranges.

Return Values

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

7.5.12.7 **IBMDSwitcherFairlightAudioEqualizerBand::GetFrequencyRange** method

The **GetFrequencyRange** method returns the current frequency range of the equalizer band.

Syntax

```
HRESULT GetFrequencyRange  
(IBMDSwitcherFairlightAudioEqualizerBand FrequencyRange* range)
```

Parameters

Name	Direction	Description
range	out	The current frequency range.

Return Values

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

7.5.12.8 **IBMDSwitcherFairlightAudioEqualizerBand::SetFrequencyRange** method

The **SetFrequencyRange** method sets the frequency range of the equalizer band.

Syntax

```
HRESULT SetFrequencyRange  
(IBMDSwitcherFairlightAudioEqualizerBand FrequencyRange range)
```

Parameters

Name	Direction	Description
range	in	The desired frequency range.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_INVALIDARG	The range is not a valid identifier.

7.5.12.9 **IBMDSwitcherFairlightAudioEqualizerBand::GetFrequencyRangeMinMax** method

The **GetFrequencyRangeMinMax** method gets the minimum and maximum frequencies of a specified **BMDSwitcherFairlightAudioEqualizerBandFrequencyRange**.

Syntax

```
HRESULT GetFrequencyRangeMinMax  
(BMDSwitcherFairlightAudioEqualizerBandFrequencyRange range,  
 uint32_t* minFreq, uint32_t* maxFreq)
```

Parameters

Name	Direction	Description
range	in	The desired frequency range.
minFreq	out	The current minimum frequency.
maxFreq	out	The current maximum frequency.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The minFreq or maxFreq parameters are invalid.
E_INVALIDARG	The range is not a valid identifier.

7.5.12.10 **IBMDSwitcherFairlightAudioEqualizerBand::GetFrequency** method

The **GetFrequency** method returns the current frequency of the equalizer band.

Syntax

```
HRESULT GetFrequency(uint32_t* freq)
```

Parameters

Name	Direction	Description
freq	out	The current frequency.

Return Values

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

7.5.12.11 **IBMDSwitcherFairlightAudioEqualizerBand::SetFrequency** method

The **SetFrequency** method sets the frequency of the equalizer band.

Syntax

```
HRESULT SetFrequency(uint32_t freq)
```

Parameters

Name	Direction	Description
freq	in	The desired frequency.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.12.12 **IBMDSwitcherFairlightAudioEqualizerBand::GetGain** method

The **GetGain** method returns the current gain of the equalizer band.

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.

7.5.12.13 **IBMDSwitcherFairlightAudioEqualizerBand::SetGain** method

The **SetGain** method sets the gain of the equalizer band.

Syntax

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

Parameters

Name	Direction	Description
gain	in	The desired gain.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.12.14 **IBMDSwitcherFairlightAudioEqualizerBand::GetQFactor** method

The **GetQFactor** method returns the current Q factor of the equalizer band.

Syntax

```
HRESULT GetQFactor(double* value)
```

Parameters

Name	Direction	Description
value	out	The current Q Factor.

Return Values

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

7.5.12.15 **IBMDSwitcherFairlightAudioEqualizerBand::SetQFactor** method

The **SetQFactor** method sets the Q factor of the equalizer band.

Syntax

```
HRESULT SetQFactor(double value)
```

Parameters

Name	Direction	Description
value	in	The desired Q Factor.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.12.16 **IBMDSwitcherFairlightAudioEqualizerBand::Reset** method

The **Reset** method resets the equalizer band to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.12.17 **IBMDSwitcherFairlightAudioEqualizerBand::AddCallback** method

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

Syntax

```
HRESULT AddCallback lightAudioEqualizerBandCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioEqualizerBandCallback object interface.

Return Values

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

7.5.12.18 **IBMDSwitcherFairlightAudioEqualizerBand::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioEqualizerBandCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioEqualizerBandCallback object interface.

Return Values

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

7.5.13 IBMDSwitcherFairlightAudioEqualizerBandCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioEqualizerBand	IID_IBMDSwitcherFairlightAudioEqualizerBand	An IBMDSwitcherFairlightAudioEqualizerBandCallback object interface is installed with IBMDSwitcherFairlightAudioEqualizerBand::AddCallback and removed with IBMDSwitcherFairlightAudioEqualizerBand::RemoveCallback

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

7.5.13.1 IBMDSwitcherFairlightAudioEqualizerBandCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioEqualizerBand** 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(IBMDSwitcherFairlightAudioEqualizerBandEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioEqualizerBandEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.14 IBMDSwitcherFairlightAudioDynamicsProcessor Interface

The `IBMDSwitcherFairlightAudioDynamicsProcessor` object interface is the root object for all Fairlight audio dynamics processing.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioMixer	IID_IBMDSwitcherFairlightAudioMixer	<code>IBMDSwitcherFairlightAudioMixer::GetMasterOutEffect</code> can return an <code>IBMDSwitcherFairlightAudioDynamicsProcessor</code> object interface.
IBMDSwitcherFairlightAudioSource	IID_IBMDSwitcherFairlightAudioSource	<code>IBMDSwitcherFairlightAudioSource::GetEffect</code> can return an <code>IBMDSwitcherFairlightAudioDynamicsProcessor</code> object interface.

Public Member Functions	
Method	Description
<code>GetProcessor</code>	Get the dynamics processor object interface.
<code>GetMakeupGain</code>	Get the current make up gain value.
<code>SetMakeupGain</code>	Set the make up gain value.
<code>Reset</code>	Reset the dynamics.
<code>ResetInputPeakLevels</code>	Reset the input peak level statistics.
<code>ResetOutputPeakLevels</code>	Reset the output peak level statistics.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.14.1 IBMDSwitcherFairlightAudioDynamicsProcessor::GetProcessor method

The `GetProcessor` method returns the dynamics processor object interface for the specified interface ID, such as `IBMDSwitcherFairlightAudioLimiter`, `IBMDSwitcherFairlightAudioCompressor`, `IBMDSwitcherFairlightAudioExpander`.

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

Syntax

```
HRESULT GetProcessor(REFIID iid, LPVOID* ppv)
```

Parameters

Name	Direction	Description
<code>iid</code>	in	Interface ID for the desired interface.
<code>ppv</code>	out	Pointer to return interface object.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>ppv</code> parameter is invalid.
<code>E_NOINTERFACE</code>	Interface was not found.

7.5.14.2 **IBMDSwitcherFairlightAudioDynamicsProcessor::GetMakeupGain** method

The **GetMakeupGain** method returns the current make up gain value.

Syntax

```
HRESULT GetMakeupGain(double* gain)
```

Parameters

Name	Direction	Description
gain	out	The current make up gain value.

Return Values

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

7.5.14.3 **IBMDSwitcherFairlightAudioDynamicsProcessor::SetMakeupGain** method

The **SetMakeupGain** method sets the make up gain value.

Syntax

```
HRESULT SetMakeupGain(double gain)
```

Parameters

Name	Direction	Description
gain	in	The desired make up gain value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.14.4 **IBMDSwitcherFairlightAudioDynamicsProcessor::Reset** method

The **Reset** method resets the dynamics to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.14.5 **IBMDSwitcherFairlightAudioDynamicsProcessor::ResetInput PeakLevels method**

The **ResetInputPeakLevels** method resets the peak input level statistics.

Syntax

```
HRESULT ResetInputPeakLevels(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.14.6 **IBMDSwitcherFairlightAudioDynamicsProcessor::ResetOutput PeakLevels method**

The **ResetOutputPeakLevels** method resets the peak output level statistics.

Syntax

```
HRESULT ResetOutputPeakLevels(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.14.7 **IBMDSwitcherFairlightAudioDynamicsProcessor::AddCallback** method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherFairlightAudioDynamicsProcessor** object. Pass an object implementing the **IBMDSwitcherFairlightAudioDynamicsProcessorCallback** interface to receive callbacks. Adding a new callback

will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioDynamicsProcessorCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioDynamics ProcessorCallback object interface.

Return Values

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

7.5.14.8 **IBMDSwitcherFairlightAudioDynamicsProcessor::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioDynamicsProcessorCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioDynamics ProcessorCallback object interface.

Return Values

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

7.5.15 IBMDSwitcherFairlightAudioDynamicsProcessorCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioDynamicsProcessor	IID_IBMDSwitcherFairlightAudioDynamicsProcessor	An IBMDSwitcherFairlightAudioDynamicsProcessorCallback object interface is installed with IBMDSwitcherFairlightAudioDynamicsProcessor::AddCallback and removed with IBMDSwitcherFairlightAudioDynamicsProcessor::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
InputLevelNotification	Reports input level statistics.
OutputLevelNotification	Reports output level statistics.

7.5.15.1 IBMDSwitcherFairlightAudioDynamicsProcessorCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioDynamicsProcessor** 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(BMDSwitcherFairlightAudioDynamics ProcessorEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherFairlightAudioDynamics ProcessorEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.15.2 IBMDSwitcherFairlightAudioDynamicsProcessorCallback::InputLevel Notification method

The **InputLevelNotification** method is called periodically to report the current dB input levels and the last known peak levels. These peak levels can be reset using **IBMSwitcherFairlightAudioDynamicsProcessor::ResetInputPeakLevels**.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 InputLevelNotification(uint32_t numLevels, const double* levels,
                                uint32_t numPeakLevels, const double* peakLevels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of input levels.
levels	in	The current input dB levels.
numPeakLevels	in	The number of input peak levels.
peakLevels	in	The highest encountered input peak dB level.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.15.3 **IBMDSwitcherFairlightAudioDynamicsProcessorCallback::OutputLevel Notification method**

The **OutputLevelNotification** method is called periodically to report the current dB output levels and the last known peak levels. These peak levels can be reset using

IBMDSwitcherFairlightAudioDynamicsProcessor::ResetOutputPeakLevels.

Note that this is an opt-in subscription. Enable or disable receiving these calls using

IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled.

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 OutputLevelNotification(uint32_t numLevels, const double* levels,
                                uint32_t numPeakLevels, const double* peakLevels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of output levels.
levels	in	The current output dB levels.
numPeakLevels	in	The number of output peak levels.
peakLevels	in	The highest encountered output peak level.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16 **IBMDSwitcherFairlightAudioLimiter Interface**

The **IBMDSwitcherFairlightAudioLimiter** object interface is used for manipulating the Fairlight audio limiter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioDynamicsProcessor	IID_IBMDSwitcherFairlightAudioDynamicsProcessor	IBMDSwitcherFairlightAudioDynamicsProcessor::GetProcessor can return an IBMDSwitcherFairlightAudioLimiter object interface.

Public Member Functions	
Method	Description
GetEnabled	Get the current limiter-enabled flag.
SetEnabled	Set the desired limiter-enabled flag.
GetThreshold	Get the current threshold value.
SetThreshold	Set the threshold value.

Public Member Functions	
Method	Description
GetAttack	Get the current attack value.
SetAttack	Set the attack value.
GetHold	Get the current hold value.
SetHold	Set the hold value.
GetRelease	Get the current release value.
SetRelease	Set the limiter release value.
Reset	Reset the limiter.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.16.1 **IBMDSwitcherFairlightAudioLimiter::GetEnabled** method

The **GetEnabled** method returns the current limiter enabled flag.

Syntax

```
HRESULT GetEnabled(boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current limiter enabled flag.

Return Values

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

7.5.16.2 **IBMDSwitcherFairlightAudioLimiter::SetEnabled** method

The **SetEnabled** method sets the limiter enabled flag.

Syntax

```
HRESULT SetEnabled(boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The desired limiter enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.3 **IBMDSwitcherFairlightAudioLimiter::GetThreshold** method

The **GetThreshold** method returns the current limiter threshold value.

Syntax

```
HRESULT GetThreshold(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current threshold value.

Return Values

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

7.5.16.4 **IBMDSwitcherFairlightAudioLimiter::SetThreshold** method

The **SetThreshold** method sets the limiter threshold value.

Syntax

```
HRESULT SetThreshold(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired threshold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.5 **IBMDSwitcherFairlightAudioLimiter::GetAttack** method

The **GetAttack** method returns the current limiter attack value.

Syntax

```
HRESULT GetAttack(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current attack value.

Return Values

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

7.5.16.6 **IBMDSwitcherFairlightAudioLimiter::SetAttack** method

The **SetAttack** method sets the limiter attack value.

Syntax

```
HRESULT SetAttack(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired attack value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.7 **IBMDSwitcherFairlightAudioLimiter::GetHold** method

The **GetHold** method returns the current limiter hold value.

Syntax

```
HRESULT GetHold(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current hold value.

Return Values

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

7.5.16.8 **IBMDSwitcherFairlightAudioLimiter::SetHold** method

The **SetHold** method sets the limiter hold value.

Syntax

```
HRESULT SetHold(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired hold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.9 **IBMDSwitcherFairlightAudioLimiter::GetRelease** method

The **GetRelease** method returns the current limiter release value.

Syntax

```
HRESULT GetRelease(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current release value.

Return Values

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

7.5.16.10 **IBMDSwitcherFairlightAudioLimiter::SetRelease** method

The **SetRelease** method sets the limiter release value.

Syntax

```
HRESULT SetRelease(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired release value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.11 **IBMDSwitcherFairlightAudioLimiter::Reset** method

The **Reset** method resets the limiter to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.16.12 **IBMDSwitcherFairlightAudioLimiter::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioLimiterCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherFairlightAudioLimiterCallback</code> object interface.

Return Values

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

7.5.16.13 **IBMDSwitcherFairlightAudioLimiter::RemoveCallback** method

The `RemoveCallback` method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioLimiterCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherFairlightAudioLimiterCallback</code> object interface.

Return Values

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

7.5.17 IBMDSwitcherFairlightAudioLimiterCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioLimiter	IID_IBMDSwitcherFairlightAudioLimiter	An IBMDSwitcherFairlightAudioLimiterCallback object interface is installed with IBMDSwitcherFairlightAudioLimiter::AddCallback and removed with IBMDSwitcherFairlightAudioLimiter::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
GainReductionLevelNotification	Reports the gain reduction level statistics.

7.5.17.1 IBMDSwitcherFairlightAudioLimiterCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioLimiter** 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(IBMDSwitcherFairlightAudioLimiterEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioLimiterEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.17.2 IBMDSwitcherFairlightAudioLimiterCallback::GainReductionLevelNotification method

The **GainReductionLevelNotification** method is called periodically to report the current gain reduction dB levels.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 GainReductionLevelNotification(uint32_t numLevels,
    const double* levels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of levels of the gain reduction.
levels	in	The current dB levels of the gain reduction.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18 IBMDSwitcherFairlightAudioCompressor Interface

The **IBMDSwitcherFairlightAudioCompressor** object interface is used for manipulating the Fairlight audio compressor.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioDynamicsProcessor	IID_IBMDSwitcherFairlightAudioDynamicsProcessor	IBMDSwitcherFairlightAudioDynamicsProcessor::GetProcessor can return an IBMDSwitcherFairlightAudioCompressor object interface.

Public Member Functions	
Method	Description
GetEnabled	Get the current limiter-enabled flag.
SetEnabled	Set the desired limiter-enabled flag.
GetThreshold	Get the current threshold value.
SetThreshold	Set the threshold value.
GetAttack	Get the current attack value.
SetAttack	Set the attack value.

Public Member Functions	
Method	Description
GetHold	Get the current hold value.
SetHold	Set the hold value.
GetRelease	Get the current release value.
SetRelease	Set the limiter release value.
Reset	Reset the limiter.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.18.1 **IBMDSwitcherFairlightAudioCompressor::GetEnabled** method

The **GetEnabled** method returns the current compressor enabled flag.

Syntax

```
HRESULT GetEnabled(boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current compressor enabled flag.

Return Values

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

7.5.18.2 **IBMDSwitcherFairlightAudioCompressor::SetEnabled** method

The **SetEnabled** method sets the desired compressor enabled flag.

Syntax

```
HRESULT SetEnabled(boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The desired compressor enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Faiure.

7.5.18.3 **IBMDSwitcherFairlightAudioCompressor::GetThreshold** method

The **GetThreshold** method returns the current compressor threshold value.

Syntax

```
HRESULT GetThreshold(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current threshold value.

Return Values

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

7.5.18.4 **IBMDSwitcherFairlightAudioCompressor::SetThreshold** method

The **SetThreshold** method sets the compressor threshold value.

Syntax

```
HRESULT SetThreshold(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired threshold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18.5 **IBMDSwitcherFairlightAudioCompressor::GetAttack** method

The **GetAttack** method returns the current compressor attack value.

Syntax

```
HRESULT GetAttack(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current attack value.

Return Values

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

7.5.18.6 **IBMDSwitcherFairlightAudioCompressor::SetAttack** method

The **SetAttack** method sets the compressor attack value.

Syntax

```
HRESULT SetAttack(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired attack value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18.7 **IBMDSwitcherFairlightAudioCompressor::GetHold** method

The **GetHold** method returns the current compressor hold value.

Syntax

```
HRESULT GetHold(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current hold value.

Return Values

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

7.5.18.8 **IBMDSwitcherFairlightAudioCompressor::SetHold** method

The **SetHold** method sets the compressor hold value.

Syntax

```
HRESULT SetHold(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired hold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18.9 **IBMDSwitcherFairlightAudioCompressor::GetRelease** method

The **GetRelease** method returns the current compressor release value.

Syntax

```
HRESULT GetRelease(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current release value.

Return Values

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

7.5.18.10 **IBMDSwitcherFairlightAudioCompressor::SetRelease** method

The **SetRelease** method sets the desired compressor release value.

Syntax

```
HRESULT SetRelease(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired release value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18.11 **IBMDSwitcherFairlightAudioCompressor::Reset** method

The **Reset** method resets the compressor to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.18.12 **IBMDSwitcherFairlightAudioCompressor::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioCompressorCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioCompressorCallback object interface.

Return Values

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

7.5.18.13 **IBMDSwitcherFairlightAudioCompressor::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioCompressorCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioCompressorCallback object interface.

Return Values

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

7.5.19 IBMDSwitcherFairlightAudioCompressorCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioCompressor	IID_IBMDSwitcherFairlightAudioCompressor	An IBMDSwitcherFairlightAudioCompressorCallback object interface is installed with IBMDSwitcherFairlightAudioCompressor::AddCallback and removed with IBMDSwitcherFairlightAudioCompressor::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
GainReductionLevelNotification	Reports the gain reduction level statistics.

7.5.19.1 IBMDSwitcherFairlightAudioCompressorCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioCompressor** 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(IBMDSwitcherFairlightAudioCompressorEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioCompressorEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.19.2 **IBMDSwitcherFairlightAudioCompressorCallback::GainReductionLevel Notification method**

The **GainReductionLevelNotification** method is called periodically to report the current gain reduction dB levels.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 GainReductionLevelNotification(uint32_t numLevels, const double* levels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of levels of the gain reduction.
levels	in	The current dB levels of the gain reduction.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20 **IBMDSwitcherFairlightAudioExpander Interface**

The **IBMDSwitcherFairlightAudioExpander** object interface is used for manipulating the Fairlight audio expander.

A reference to an **IBMDSwitcherFairlightAudioExpander** object interface may be obtained from an **IBMDSwitcherFairlightAudioDynamicsProcessor** object interface using the **GetProcessor** method.

Pass **IID_IBMDSwitcherFairlightAudioExpander** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioDynamicsProcessor	IID_IBMDSwitcherFairlightAudioDynamicsProcessor	IBMDSwitcherFairlightAudioDynamicsProcessor::GetProcessor can return an IBMDSwitcherFairlightAudioExpander object interface.

Public Member Functions	
Method	Description
GetEnabled	Get the current expander-enabled flag.
SetEnabled	Set the desired expander-enabled flag.
GetGateMode	Get the current gate mode.
SetGateMode	Set the gate mode.
GetThreshold	Get the current threshold value.
SetThreshold	Set the threshold value.

Public Member Functions	
Method	Description
GetRange	Get the current range value.
SetRange	Set the range value.
GetRatio	Get the current ratio value.
SetRatio	Set the ratio value.
GetAttack	Get the current attack value.
SetAttack	Set the attack value.
GetHold	Get the hold value.
SetHold	Set the hold value.
GetRelease	Get the current release value.
SetRelease	Set the release value.
Reset	Reset the expander.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.20.1 **IBMDSwitcherFairlightAudioExpander::GetEnabled** method

The **GetEnabled** method returns the current expander enabled flag.

Syntax

```
HRESULT GetEnabled(boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current expander enabled flag.

Return Values

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

7.5.20.2 **IBMDSwitcherFairlightAudioExpander::SetEnabled** method

The **SetEnabled** method sets the expander enabled flag.

Syntax

```
HRESULT SetEnabled(boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The desired expander enabled flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.3 **IBMDSwitcherFairlightAudioExpander::GetGateMode** method

The **GetGateMode** method returns the current expander gate mode.

Syntax

```
HRESULT GetGateMode(boolean* gateMode)
```

Parameters

Name	Direction	Description
gateMode	out	The current gate mode.

Return Values

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

7.5.20.4 **IBMDSwitcherFairlightAudioExpander::SetGateMode** method

The **SetGateMode** method sets the expander gate mode.

Syntax

```
HRESULT SetGateMode(boolean gateMode)
```

Parameters

Name	Direction	Description
gateMode	in	The desired gate mode.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.5 **IBMDSwitcherFairlightAudioExpander::GetThreshold** method

The **GetThreshold** method returns the current expander threshold value.

Syntax

```
HRESULT GetThreshold(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current threshold value.

Return Values

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

7.5.20.6 **IBMDSwitcherFairlightAudioExpander::SetThreshold** method

The **SetThreshold** method sets the expander threshold value.

Syntax

```
HRESULT SetThreshold(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired threshold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.7 **IBMDSwitcherFairlightAudioExpander::GetRange** method

The **GetRange** method returns the current expander range value.

Syntax

```
HRESULT GetRange(double* range)
```

Parameters

Name	Direction	Description
range	out	The current range value.

Return Values

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

7.5.20.8 **IBMDSwitcherFairlightAudioExpander::SetRange** method

The **SetRange** method sets the expander range value.

Syntax

```
HRESULT SetRange(double range)
```

Parameters

Name	Direction	Description
range	in	The desired range value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.9 **IBMDSwitcherFairlightAudioExpander::GetRatio** method

The **GetRatio** method returns the current expander ratio value.

Syntax

```
HRESULT GetRatio(double* ratio)
```

Parameters

Name	Direction	Description
ratio	out	The current ratio value.

Return Values

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

7.5.20.10 **IBMDSwitcherFairlightAudioExpander::SetRatio** method

The **SetRatio** method sets the expander ratio value.

Syntax

```
HRESULT SetRatio(double ratio)
```

Parameters

Name	Direction	Description
ratio	in	The desired ratio value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.11 **IBMDSwitcherFairlightAudioExpander::GetAttack** method

The **GetAttack** method returns the current expander attack value.

Syntax

```
HRESULT GetAttack(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current attack value.

Return Values

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

7.5.20.12 **IBMDSwitcherFairlightAudioExpander::SetAttack** method

The **SetAttack** method sets the expander attack value.

Syntax

```
HRESULT SetAttack(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired attack value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.13 **IBMDSwitcherFairlightAudioExpander::GetHold** method

The **GetHold** method returns the current expander hold value.

Syntax

```
HRESULT GetHold(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current hold value.

Return Values

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

7.5.20.14 **IBMDSwitcherFairlightAudioExpander::SetHold** method

The **SetHold** method sets the expander hold value.

Syntax

```
HRESULT SetHold(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired hold value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.15 **IBMDSwitcherFairlightAudioExpander::GetRelease** method

The **GetRelease** method returns the current expander release value.

Syntax

```
HRESULT GetRelease(double* ms)
```

Parameters

Name	Direction	Description
ms	out	The current release value.

Return Values

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

7.5.20.16 **IBMDSwitcherFairlightAudioExpander::SetRelease** method

The **SetRelease** method sets the expander release value.

Syntax

```
HRESULT SetRelease(double ms)
```

Parameters

Name	Direction	Description
ms	in	The desired release value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.17 **IBMDSwitcherFairlightAudioExpander::Reset** method

The **Reset** method resets the expander to its default state.

Syntax

```
HRESULT Reset(void)
```

Parameters

none.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.20.18 **IBMDSwitcherFairlightAudioExpander::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioExpanderCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioExpanderCallback object interface.

Return Values

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

7.5.20.19 IBMDSwitcherFairlightAudioExpander::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioExpanderCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioExpander Callback object interface.

Return Values

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

7.5.21 IBMDSwitcherFairlightAudioExpanderCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioExpander	IID_IBMDSwitcherFairlightAudioExpander	An IBMDSwitcherFairlightAudioExpanderCallback object interface is installed with IBMDSwitcherFairlightAudioExpander::AddCallback and removed with IBMDSwitcherFairlightAudioExpander::RemoveCallback

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
GainReductionLevelNotification	Reports the gain reduction level statistics.

7.5.21.1 **IBMDSwitcherFairlightAudioExpanderCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherFairlightAudioExpander** 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(BMDSwitcherFairlightAudioExpanderEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherFairlightAudioExpander EventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.21.2 **IBMDSwitcherFairlightAudioExpanderCallback::GainReductionLevel** Notification method

The **GainReductionLevelNotification** method is called periodically to report the current gain reduction dB levels.

Note that this is an opt-in subscription. Enable or disable receiving these calls using **IBMDSwitcherFairlightAudioMixer::SetAllLevelNotificationsEnabled**.

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 GainReductionLevelNotification(uint32_t numLevels, const double* levels)
```

Parameters

Name	Direction	Description
numLevels	in	The number of levels of the gain reduction.
levels	in	The current dB levels of the gain reduction.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.22 IBMDSwitcherFairlightAudioHeadphoneOutputIterator Interface

The `IBMDSwitcherFairlightAudioHeadphoneOutputIterator` is used to enumerate the available headphone outputs for the Fairlight audio mixer.

A reference to an `IBMDSwitcherFairlightAudioHeadphoneOutputIterator` object interface may be obtained from an `IBMDSwitcherFairlightAudioMixer` object interface using the `CreateIterator` method.

Pass `IID_IBMDSwitcherFairlightAudioHeadphoneOutputIterator` for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioMixer</code>	<code>IID_IBMDSwitcherFairlightAudioMixer</code>	<code>IBMDSwitcherFairlightAudioMixer::CreateIterator</code> can return an <code>IBMDSwitcherFairlightAudioHeadphoneOutputIterator</code> object interface.

Public Member Functions	
Method	Description
<code>Next</code>	Returns a pointer to the next <code>IBMDSwitcherFairlightAudioHeadphoneOutput</code> object interface.

7.5.22.1 IBMDSwitcherFairlightAudioHeadphoneOutputIterator::Next method

The `Next` method returns the next available `IBMDSwitcherFairlightAudioHeadphoneOutput` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioHeadphoneOutput* audioHeadphoneOutput)
```

Parameters

Name	Direction	Description
<code>audioHeadphoneOutput</code>	out	<code>IBMDSwitcherFairlightAudioHeadphoneOutput</code> object interface.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>S_FALSE</code>	No more <code>IBMDSwitcherFairlightAudioHeadphoneOutput</code> objects available.
<code>E_POINTER</code>	The <code>audioHeadphoneOutput</code> parameter is invalid.

7.5.23 IBMDSwitcherFairlightAudioHeadphoneOutput Interface

The `IBMDSwitcherFairlightAudioHeadphoneOutput` object interface is used for manipulating parameters specific to Fairlight audio headphone outputs.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioHeadphoneOutputIterator	IID_IBMDSwitcherFairlightAudioHeadphoneOutputIterator	An <code>IBMDSwitcherFairlightAudioHeadphoneOutput</code> interface can be obtained with <code>IBMDSwitcherFairlightAudioHeadphoneOutputIterator::Next</code>

Public Member Functions	
Method	Description
DoesSupportSolo	Determines if solo monitoring is supported by the switcher.
DoesSupportMute	Determines if mute is supported by the switcher.
GetGain	Get the current gain value.
SetGain	Set the gain value.
GetInputMasterOutGain	Get the current master out input gain value.
SetInputMasterOutGain	Set the master out input gain value.
GetInputMasterOutMute	Get the mute state of the master output.
SetInputMasterOutMute	Set the mute state of the master output.
DoesSupportTalkback	Determines if talkback is supported by the switcher.
GetInputTalkbackGain	Get the current talkback input gain value.
SetInputTalkbackGain	Set the talkback input gain value.
GetInputTalkbackMute	Get the mute state of the talkback output.
SetInputTalkbackMute	Set the mute state of the talkback output.
DoesSupportSidetone	Determines if sidetone is supported by the switcher.
GetInputSidetoneGain	Get the current headphone sidetone (microphone) input gain value.
SetInputSidetoneGain	Set the headphone sidetone (microphone) input gain value.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

7.5.23.1 **IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportSolo** method

The **DoesSupportSolo** method is used to determine whether solo monitoring is supported by the switcher.

Syntax

```
HRESULT DoesSupportSolo(Boolean* supportsSolo)
```

Parameters

Name	Direction	Description
supportsSolo	out	Boolean value describing whether solo monitoring is supported by the switcher.

Return Values

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

7.5.23.2 **IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportMute** method

The **DoesSupportMute** method is used to determine whether mute is supported by the switcher.

Syntax

```
HRESULT DoesSupportMute(Boolean* supportsMute)
```

Parameters

Name	Direction	Description
supportsMute	out	Boolean value describing whether mute is supported by the switcher.

Return Values

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

7.5.23.3 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetGain** method

The **GetGain** method returns the current gain applied to the Fairlight audio headphone output.

Syntax

```
HRESULT GetGain(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current decibel value.

Return Values

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

7.5.23.4 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetGain** method

The **SetGain** method sets the gain to apply to the Fairlight headphone output.

Syntax

```
HRESULT SetGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired decibel value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.23.5 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputMasterOutGain** method

The **GetInputMasterOutGain** method returns the gain currently applied to the master out input of the headphone output.

Syntax

```
HRESULT GetInputMasterOutGain(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current decibel value.

Return Values

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

7.5.23.6 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputMasterOutGain** method

The **SetInputMasterOutGain** method sets the gain to apply to the master out input of the headphone output.

Syntax

```
HRESULT SetInputMasterOutGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired decibel value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.23.7 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputMasterOutMute** method

The **GetInputMasterOutMute** method is used to determine the current mute state of the master output.

Syntax

```
HRESULT GetInputMasterOutMute(Boolean* muteMaster)
```

Parameters

Name	Direction	Description
muteMaster	out	The mute state of the master output.

Return Values

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

7.5.23.8 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputMasterOutMute** method

The **SetInputMasterOutMute** method sets the mute state of the master output.

Syntax

```
HRESULT SetInputMasterOutMute(Boolean muteMaster)
```

Parameters

Name	Direction	Description
muteMaster	in	The mute state of the master output.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support master output mute.
E_FAIL	Failure.

7.5.23.9 **IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportTalkback** method

The **DoesSupportTalkback** method is used to determine whether talkback is supported by the switcher.

Syntax

HRESULT DoesSupportTalkback(Boolean* supportsTalkback)

Parameters

Name	Direction	Description
supportsTalkback	out	Boolean value describing whether talkback is supported by the switcher.

Return Values

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

7.5.23.10 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputTalkbackGain** method

The **GetInputTalkbackGain** method returns the gain currently applied to the talkback input of the headphone output.

Syntax

HRESULT GetInputTalkbackGain(double* decibel)

Parameters

Name	Direction	Description
decibel	out	The current decibel value.

Return Values

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

7.5.23.11 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputTalkbackGain** method

The **SetInputTalkbackGain** method sets the gain to apply to the talkback input of the headphone output.

Syntax

```
HRESULT SetInputTalkbackGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired decibel value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.23.12 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputTalkbackMute** method

The **GetInputTalkbackMute** method is used to determine the current mute state of the talkback output.

Syntax

```
HRESULT GetInputTalkbackMute(BOOLEAN* muteTalkback)
```

Parameters

Name	Direction	Description
muteTalkback	out	The mute state of the talkback output.

Return Values

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

7.5.23.13 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputTalkbackMute** method

The **SetInputTalkbackMute** method sets the mute state of the talkback output.

Syntax

```
HRESULT SetInputTalkbackMute(Boolean muteTalkback)
```

Parameters

Name	Direction	Description
muteTalkback	in	The mute state of the talkback output.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support talkback mute.
E_FAIL	Failure.

7.5.23.14 **IBMDSwitcherFairlightAudioHeadphoneOutput::DoesSupportSidetone** method

The **DoesSupportSidetone** method is used to determine whether sidetone is supported by the switcher..

Syntax

```
HRESULT DoesSupportSidetone(Boolean* supportsSidetone)
```

Parameters

Name	Direction	Description
supportsSidetone	out	Boolean value describing whether sidetone is supported by the switcher.

Return Values

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

7.5.23.15 **IBMDSwitcherFairlightAudioHeadphoneOutput::GetInputSidetoneGain** method

The **GetInputSidetoneGain** method returns the gain currently applied to the sidetone (microphone) input of the headphone output.

Syntax

```
HRESULT GetInputSidetoneGain(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The current decibel value.

Return Values

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

7.5.23.16 **IBMDSwitcherFairlightAudioHeadphoneOutput::SetInputSidetoneGain** method

The **SetInputSidetoneGain** method sets the gain to apply to the sidetone (microphone) input of the headphone output.

Syntax

```
HRESULT SetInputSidetoneGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The desired decibel value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.23.17 **IBMDSwitcherFairlightAudioHeadphoneOutput::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioHeadphoneOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioHeadphone OutputCallback object interface.

Return Values

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

7.5.23.18 **IBMDSwitcherFairlightAudioHeadphoneOutput::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioHeadphoneOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioHeadphone OutputCallback object interface.

Return Values

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

7.5.24 IBMDSwitcherFairlightAudioHeadphoneOutputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioHeadphoneOutput	IID_IBMDSwitcherFairlightAudioHeadphoneOutput	An IBMDSwitcherFairlightAudioHeadphoneOutputCallback object interface is installed with IBMDSwitcherFairlightAudioHeadphoneOutput::AddCallback and removed with IBMDSwitcherFairlightAudioHeadphoneOutput::RemoveCallback

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

7.5.24.1 IBMDSwitcherFairlightAudioHeadphoneOutputCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioHeadphoneOutput** 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.

Syntax

```
HRESULT Notify(IBMDSwitcherFairlightAudioHeadphoneOutputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherFairlightAudioHeadphone OutputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.25 IBMDSwitcherFairlightAnalogAudioInput Interface

The `IBMDSwitcherFairlightAnalogAudioInput` object interface is used for manipulating Fairlight audio input settings specific to analog inputs.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioInput</code>	<code>IID_IBMDSwitcherFairlightAudioInput</code>	An <code>IBMDSwitcherFairlightAnalogAudioInput</code> object interface can be obtained with <code>IBMDSwitcherFairlightAudioInput::QueryInterface</code> .

Public Member Functions	
Method	Description
<code>GetSupportedInputLevels</code>	Get the supported analog input levels.
<code>GetInputLevel</code>	Get the current analog input level.
<code>SetInputLevel</code>	Set the analog input level.
<code>GetSupportedMicPowerModes</code>	Get the supported microphone power modes.
<code>GetMicPowerMode</code>	Get the current microphone power mode.
<code>SetMicPowerMode</code>	Set the microphone power mode.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.25.1 IBMDSwitcherFairlightAnalogAudioInput::GetSupportedInputLevels method

The `GetSupportedInputLevels` method gets the available analog input levels for this audio input, given as a bit mask of `BMDSwitcherFairlightAudioAnalogInputLevel`. This bit mask can be bitwise-ANDed with any value of `BMDSwitcherFairlightAudioAnalogInputLevel` (e.g. `bmdSwitcherFairlightAudioAnalogInputLevelIMicrophone`) to determine if that input level is available for this input.

Syntax

```
HRESULT GetSupportedInputLevels(BMDSwitcherFairlightAudioAnalogInputLevel* levels)
```

Parameters

Name	Direction	Description
<code>levels</code>	out	The available input levels for this audio input as a bit mask of <code>BMDSwitcherFairlightAudioAnalogInputLevel</code> .

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>levels</code> parameter is invalid.

7.5.25.2 **IBMDSwitcherFairlightAnalogAudioInput::GetInputLevel** method

The **GetInputLevel** method gets the current input level setting of the audio input. This may change if the input supports multiple input levels, generating the event **bmdSwitcherFairlightAnalogAudioInputEventTypeLevelChanged**.

Syntax

```
HRESULT GetInputLevel(BMDSwitcherFairlightAudioAnalogInputLevel* level)
```

Parameters

Name	Direction	Description
level	out	The current input level.

Return Values

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

7.5.25.3 **IBMDSwitcherFairlightAnalogAudioInput::SetInputLevel** method

The **SetInputLevel** method sets the input level for this input using a **BMDSwitcherFairlightAudioAnalogInputLevel**. Not all input levels are supported for a given audio input. Call the **GetSupportedInputLevels** method to determine the available input levels for this audio input.

Syntax

```
HRESULT SetInputLevel(BMDSwitcherFairlightAudioAnalogInputLevel level)
```

Parameters

Name	Direction	Description
level	in	The desired input level.

Return Values

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

7.5.25.4 **IBMDSwitcherFairlightAnalogAudioInput::GetSupportedMicPowerModes** method

The **GetSupportedMicPowerModes** method gets the available microphone power modes for this analog audio input, given as a bit mask of **BMDSwitcherFairlightAudioAnalogInputMicPowerMode**. This bit mask can be bitwise-ANDed with any value of **BMDSwitcherFairlightAudioAnalogInputMicPowerMode** (e.g. **bmdSwitcherFairlightAudioAnalogInputMicPowerModePlugInPower**) to determine if that power mode is available for this input.

Syntax

```
HRESULT GetSupportedMicPowerModes  
(BMDSwitcherFairlightAudioAnalogInputMicPowerMode* powerModes)
```

Parameters

Name	Direction	Description
powerModes	out	The available microphone power modes for this analog audio input as a bit mask of BMDSwitcherFairlightAudioAnalogInputMicPowerMode .

Return Values

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

7.5.25.5 **IBMDSwitcherFairlightAnalogAudioInput::GetMicPowerMode** method

The **GetMicPowerMode** method gets the current microphone power mode setting of the analog audio input. This may change if the input supports multiple power modes, generating the event **bmdSwitcherFairlightAnalogAudioInputEventTypeMicrophonePowerModeChanged**.

Syntax

```
HRESULT GetMicPowerMode  
(BMDSwitcherFairlightAudioAnalogInputMicPowerMode* powerMode)
```

Parameters

Name	Direction	Description
powerMode	out	The current microphone power mode for this input.

Return Values

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

7.5.25.6 **IBMDSwitcherFairlightAnalogAudioInput::SetMicPowerMode** method

The **SetMicPowerMode** method sets the microphone power mode for this input using a **BMDSwitcherFairlightAudioAnalogInputMicPowerMode**. Not all microphone power modes are supported for a given audio input. Call the **GetSupportedMicrophonePowerModes** method to determine the available power modes for this audio input. The power mode is only active if the input levels for this input are set to **bmdSwitcherFairlightAudioAnalogInputLevelMicrophone**.

Syntax

```
HRESULT SetMicPowerMode  
(BMDSwitcherFairlightAudioAnalogInputMicPowerMode powerMode)
```

Parameters

Name	Direction	Description
powerMode	in	The desired microphone power mode.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The powerMode parameter is not a valid microphone power mode for this input.
E_FAIL	Failure.

7.5.25.7 **IBMDSwitcherFairlightAnalogAudioInput::AddCallback** method

The **AddCallback** method configures a callback to be called when a Fairlight analog audio input property changes, such as a change in the input level.

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(IBMDSwitcherFairlightAnalogAudioInputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAnalogAudioInputCallback interface.

Return Values

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

7.5.25.8 IBMDSwitcherFairlightAnalogAudioInput::RemoveCallback method

The RemoveCallback method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAnalogAudioInputCallback* 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.

7.5.26 IBMDSwitcherFairlightAnalogAudioInputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAnalogAudioInput	IID_IBMDSwitcherFairlightAnalogAudioInput	An IBMDSwitcherFairlightAnalogAudioInputCallback object interface is installed with IBMDSwitcherFairlightAnalogAudioInput::AddCallback and removed with IBMDSwitcherFairlightAnalogAudioInput::RemoveCallback .

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

7.5.26.1 **IBMDSwitcherFairlightAnalogAudioInputCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherFairlightAnalogAudioInput** 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(IBMDSwitcherFairlightAnalogAudioInputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAnalogAudioInputEventType that describes the type of event that occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.27 **IBMDSwitcherFairlightAudioSolo** Interface

The **IBMDSwitcherFairlightAudioSolo** object interface is used for manipulating Fairlight audio output monitoring settings.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioHeadphoneOutput	IID_IBMDSwitcherFairlightAudioHeadphoneOutput	An IBMDSwitcherFairlightAudioSolo object interface can be obtained with IBMDSwitcherFairlightAudioHeadphoneOutput::QueryInterface .

Public Member Functions	
Method	Description
GetSolo	Get the current solo state.
SetSolo	Set the solo state.
GetSoloInput	Get the ID of the current solo input.
SetSoloInput	Set the ID of the solo input.
AddCallback	Add a callback
RemoveCallback	Remove a callback

7.5.27.1 IBMDSwitcherFairlightAudioSolo::GetSolo method

The `GetSolo` method gets the solo state of the audio output monitor.

Syntax

```
HRESULT GetSolo(Boolean* solo)
```

Parameters

Name	Direction	Description
solo	out	The current solo state of the audio monitor.

Return Values

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

7.5.27.2 IBMDSwitcherFairlightAudioSolo::SetSolo method

The `SetSolo` method sets the solo state of the audio output monitor.

Syntax

```
HRESULT SetSolo(Boolean solo)
```

Parameters

Name	Direction	Description
solo	in	The desired solo state of the audio monitor.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.27.3 **BMDSwitcherFairlightAudioSolo::GetSoloInput** method

The **GetSoloInput** method gets the input ID and source ID of the audio source being monitored.

Syntax

```
HRESULT GetSoloInput(BMDSwitcherAudioInputId* audioInput,  
BMDSwitcherFairlightAudioSourceId* audioSource)
```

Parameters

Name	Direction	Description
audioInput	out	BMDSwitcherAudioInputId of the current solo audio input.
audioSource	out	BMDSwitcherFairlightAudioSourceId of the current solo audio source.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

7.5.27.4 **BMDSwitcherFairlightAudioSolo::SetSoloInput** method

The **SetSoloInput** method sets the input ID and source ID of the audio source to be monitored.

Syntax

```
HRESULT SetSoloInput(BMDSwitcherAudioInputId audioInput,  
BMDSwitcherFairlightAudioSourceId audioSource)
```

Parameters

Name	Direction	Description
audioInput	in	BMDSwitcherAudioInputId of the desired solo audio input.
audioSource	in	BMDSwitcherFairlightAudioSourceId of the desired solo audio source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.27.5 IBMDSwitcherFairlightAudioSolo::AddCallback method

The **AddCallback** method configures a callback to be called when events occur for an **IBMDSwitcherFairlightAudioSolo** object. Pass an object implementing the **IBMDSwitcherFairlightAudioSoloCallback** interface to receive callbacks.

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioSoloCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioSoloCallback object interface.

Return Values

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

7.5.27.6 IBMDSwitcherFairlightAudioSolo::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioSoloCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioSoloCallback object interface.

Return Values

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

7.5.28 IBMDSwitcherFairlightAudioSoloCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioSolo	IID_ IBMDSwitcherFairlightAudioSolo	An IBMDSwitcherFairlightAudioSoloCallback object interface is installed with IBMDSwitcherFairlightAudioSolo::AddCallback and removed with IBMDSwitcherFairlightAudioSolo::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.

7.5.28.1 IBMDSwitcherFairlightAudioSoloCallback::Notify method

The **Notify** method is called when **IBMDSwitcherFairlightAudioSolo** events occur, such as solo state or solo input 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(IBMDSwitcherFairlightAudioSoloEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherFairlightAudioSoloEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.29 IBMDSwitcherFairlightAudioAuxOutputIterator Interface

The **IBMDSwitcherFairlightAudioAuxOutputIterator** object interface is used to enumerate the available auxiliary audio outputs for the Fairlight audio mixer.

A reference to an **IBMDSwitcherFairlightAudioAuxOutputIterator** object interface may be obtained from an **IBMDSwitcherFairlightAudioMixer** object interface using the **CreateIterator** method. Pass **IID_IBMDSwitcherFairlightAudioInputIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioMixer	ID_IBMDSwitcherFairlightAudioMixer	IBMDSwitcherFairlightAudioMixer::CreateIterator can return an IBMDSwitcherFairlightAudioAuxOutputIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherFairlightAudioAuxOutput object interface.

7.5.29.1 IBMDSwitcherFairlightAudioAuxOutputIterator::Next method

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

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

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioAuxOutput** audioAuxOutput)
```

Parameters

Name	Direction	Description
audioAuxOutput	out	IBMDSwitcherFairlightAudioAuxOutput object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherFairlightAudioAuxOutput objects available.
E_POINTER	The audioAuxOutput parameter is invalid.

7.5.30 IBMDSwitcherFairlightAudioAuxOutput Interface

The `IBMDSwitcherFairlightAudioAuxOutput` object interface is used for managing the Fairlight audio auxiliary outputs, such as Studio and Control audio outputs.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioAuxOutputIterator</code>	<code>IID_IBMDSwitcherFairlightAudioAuxOutputIterator</code>	An <code>IBMDSwitcherFairlightAudioAuxOutput</code> object interface will be returned after a successful call to <code>IBMDSwitcherFairlightAudioAuxOutputIterator::Next</code> method.

Public Member Functions	
Method	Description
<code>GetOutputId</code>	Get the unique ID for this auxiliary audio output.
<code>DoesSupportMute</code>	Determine if mute is supported.
<code>DoesSupportDim</code>	Determine if dim is supported.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set the gain value.
<code>GetDimOn</code>	Get the current dim state.
<code>SetDimOn</code>	Set the dim state.
<code>GetDimLevel</code>	Get the current dim level.
<code>SetDimLevel</code>	Set the dim level.
<code>GetMute</code>	Get the current mute state.
<code>SetMute</code>	Set the mute state.
<code>CreateIterator</code>	Create an iterator.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.30.1 IBMDSwitcherFairlightAudioAuxOutput::GetOutputId method

The `GetOutputId` method returns the unique ID for the auxiliary audio output.

Syntax

```
HRESULT GetOutputId(IBMDSwitcherFairlightAudioAuxOutputId* outputId)
```

Parameters

Name	Direction	Description
<code>outputId</code>	out	Unique ID for the audio output.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>outputId</code> parameter is invalid.

7.5.30.2 **IBMDSwitcherFairlightAudioAuxOutput::DoesSupportMute** method

The **DoesSupportMute** method is used to determine whether mute is supported by this auxiliary audio output.

Syntax

```
HRESULT DoesSupportMute(Boolean* supportsMute)
```

Parameters

Name	Direction	Description
supportsMute	out	Boolean value describing whether mute is supported by this auxiliary audio output.

Return Values

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

7.5.30.3 **IBMDSwitcherFairlightAudioAuxOutput::DoesSupportDim** method

The **DoesSupportDim** method is used to determine whether dim is supported by this auxiliary audio output.

Syntax

```
HRESULT DoesSupportDim(Boolean* supportsDim)
```

Parameters

Name	Direction	Description
supportsDim	out	Boolean value describing whether dim is supported by this auxiliary audio output.

Return Values

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

7.5.30.4 **IBMDSwitcherFairlightAudioAuxOutput::GetGain** method

The **GetGain** method returns the current gain value applied to the auxiliary audio output.

Syntax

```
HRESULT GetGain(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The gain currently applied to the auxiliary audio output.

Return Values

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

7.5.30.5 IBMDSwitcherFairlightAudioAuxOutput::SetGain method

The **SetGain** method sets the gain value to apply to the auxiliary audio output.

Syntax

```
HRESULT SetGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The gain to apply to the auxiliary audio output.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.30.6 IBMDSwitcherFairlightAudioAuxOutput::GetDimOn method

The **GetDimOn** method is used to determine whether the auxiliary audio output is currently dimmed.

Syntax

```
HRESULT GetDimOn(Boolean* dimOn)
```

Parameters

Name	Direction	Description
dimOn	out	The current dim state of the auxiliary audio output.

Return Values

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

7.5.30.7 IBMDSwitcherFairlightAudioAuxOutput::SetDimOn method

The **SetDimOn** method is used to enable or disable dim on the auxiliary audio output.

Syntax

```
HRESULT SetDimOn(Boolean dimOn)
```

Parameters

Name	Direction	Description
dimOn	in	The dim state to apply to the auxiliary audio output.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.30.8 IBMDSwitcherFairlightAudioAuxOutput::GetDimLevel method

The `GetDimLevel` method returns the current dim level value applied to the auxiliary audio output.

Syntax

```
HRESULT GetDimLevel(double* dimLevel)
```

Parameters

Name	Direction	Description
dimLevel	out	The dim level currently applied to the auxiliary audio output.

Return Values

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

7.5.30.9 IBMDSwitcherFairlightAudioAuxOutput::SetDimLevel method

The `SetDimLevel` method sets the dim level value to apply to the auxiliary audio output.

Syntax

```
HRESULT SetDimLevel(double dimLevel)
```

Parameters

Name	Direction	Description
dimLevel	in	The dim level to apply to the auxiliary audio output.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.30.10 IBMDSwitcherFairlightAudioAuxOutput::GetMute method

The `GetMute` method is used to determine whether the auxiliary audio output is currently muted.

Syntax

```
HRESULT GetMute(Boolean* mute)
```

Parameters

Name	Direction	Description
mute	out	The current mute state of the auxiliary audio output.

Return Values

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

7.5.30.11 **IBMDSwitcherFairlightAudioAuxOutput::SetMute** method

The **SetMute** method is used to enable or disable mute on the auxiliary audio output.

Syntax

```
HRESULT SetMute(Boolean mute)
```

Parameters

Name	Direction	Description
mute	in	The mute state to apply to the auxiliary audio output.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.30.12 **IBMDSwitcherFairlightAudioAuxOutput::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	out	Pointer to the returned 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.

7.5.30.13 **IBMDSwitcherFairlightAudioAuxOutput::AddCallback** method

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

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

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioAuxOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioAuxOutput object interface.

Return Values

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

7.5.30.14 **IBMDSwitcherFairlightAudioAuxOutput::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioAuxOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioAuxOutput object interface.

Return Values

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

7.5.31 **IBMDSwitcherFairlightAudioAuxOutputCallback** Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioAuxOutput	IID_ IBMDSwitcherFairlightAudioAuxOutput	An IBMDSwitcherFairlightAudioAuxOutputCallback object interface is installed with IBMDSwitcherFairlightAudioAuxOutput::AddCallback and removed with IBMDSwitcherFairlightAudioAuxOutput::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.

7.5.31.1 **IBMDSwitcherFairlightAudioAuxOutputCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherFairlightAudioMixer** 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(BMDSwitcherFairlightAudioAuxOutputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherFairlightAudioAuxOutputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.5.32 IBMDSwitcherFairlightAudioAuxOutputInputIterator Interface

The **IBMDSwitcherFairlightAudioAuxOutputInputIterator** is used to enumerate the audio inputs available for routing to the audio auxiliary outputs for the Fairlight audio mixer.

A reference to an **IBMDSwitcherFairlightAudioAuxOutputInputIterator** object interface may be obtained from an **IBMDSwitcherFairlightAudioAuxOutput** object interface using the **CreateIterator** method. Pass **IID_IBMDSwitcherFairlightAudioAuxOutputInputIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherFairlightAudioAuxOutput	IID_IBMDSwitcherFairlightAudioAuxOutput	IBMDSwitcherFairlightAudioAuxOutput::CreateIterator can return an IBMDSwitcherFairlightAudioAuxOutputInputIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherFairlightAudioAuxOutputInput object interface.

7.5.32.1 IBMDSwitcherFairlightAudioAuxOutputInputIterator::Next method

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

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

Syntax

```
HRESULT Next(IBMDSwitcherFairlightAudioAuxOutputInput** auxOutputInput)
```

Parameters

Name	Direction	Description
auxOutputInput	out	IBMDSwitcherFairlightAudioAuxOutputInput object interface.

Return Values

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

7.5.33 IBMDSwitcherFairlightAudioAuxOutputInput Interface

The `IBMDSwitcherFairlightAudioAuxOutputInput` object interface is used for managing audio inputs for the Fairlight audio auxiliary outputs, such as Studio and Control audio outputs.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherFairlightAudioAuxOutputInputIterator</code>	<code>IID_IBMDSwitcherFairlightAudioAuxOutputInputIterator</code>	An <code>IBMDSwitcherFairlightAudioAuxOutputInput</code> object interface will be returned after a successful call to <code>IBMDSwitcherFairlightAudioSourceIterator::Next</code> method.

Public Member Functions	
Method	Description
<code>GetInputId</code>	Get the unique ID for this audio input.
<code>DoesSupportMute</code>	Determine if mute is supported.
<code>GetGain</code>	Get the current gain value.
<code>SetGain</code>	Set the gain value.
<code>GetMute</code>	Get the current mute state.
<code>SetMute</code>	Set the mute state.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.5.33.1 IBMDSwitcherFairlightAudioAuxOutputInput::GetInputId method

The `GetInputId` method returns the unique ID for the audio source.

Syntax

```
HRESULT GetInputId(IBMDSwitcherFairlightAudioAuxOutputInputId* inputId)
```

Parameters

Name	Direction	Description
<code>inputId</code>	out	Unique ID for the audio input.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>inputId</code> parameter is invalid.

7.5.33.2 **IBMDSwitcherFairlightAudioAuxOutputInput::DoesSupportMute** method

The **DoesSupportMute** method is used to determine whether mute is supported by this audio input.

Syntax

```
HRESULT DoesSupportMute(Boolean* supportsMute)
```

Parameters

Name	Direction	Description
supportsMute	out	Boolean value describing whether mute is supported by this audio input.

Return Values

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

7.5.33.3 **IBMDSwitcherFairlightAudioAuxOutputInput::GetGain** method

The **GetGain** method returns the current gain value applied to the audio input for the auxiliary audio output.

Syntax

```
HRESULT GetGain(double* decibel)
```

Parameters

Name	Direction	Description
decibel	out	The gain currently applied to the audio input.

7.5.33.4 **IBMDSwitcherFairlightAudioAuxOutputInput::SetGain** method

The **SetGain** method sets the gain value to apply to the audio input for the auxiliary audio output.

Syntax

```
HRESULT SetGain(double decibel)
```

Parameters

Name	Direction	Description
decibel	in	The gain to apply to the audio input.

7.5.33.5 IBMDSwitcherFairlightAudioAuxOutputInput::GetMute method

The **GetMute** method is used to determine whether the audio input is muted on the auxiliary audio output.

Syntax

```
HRESULT GetMute(Boolean* mute)
```

Parameters

Name	Direction	Description
mute	out	The current mute state of the audio input.

Return Values

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

7.5.33.6 IBMDSwitcherFairlightAudioAuxOutputInput::SetMute method

The **SetMute** method is used to enable or disable mute for the audio input on the auxiliary audio output.

Syntax

```
HRESULT SetMute(Boolean mute)
```

Parameters

Name	Direction	Description
mute	in	The mute state to apply to the audio input.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The audio input does not support mute.
E_FAIL	Failure.

7.5.33.7 **IBMDSwitcherFairlightAudioAuxOutputInput::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherFairlightAudioAuxOutputInputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioAuxOutputInputCallback object interface.

Return Values

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

7.5.33.8 **IBMDSwitcherFairlightAudioAuxOutputInput::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherFairlightAudioAuxOutputInputCallback*  
callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherFairlightAudioAuxOutputInputCallback object interface.

Return Values

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

7.5.34 IBMDSwitcherFairlightAudioAuxOutputInputCallback Interface

The `IBMDSwitcherFairlightAudioAuxOutputInputCallback` object interface is a callback class containing methods that are called when an event occurs on an `IBMDSwitcherFairlightAudioAuxOutputInput` object.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
<code>BMDSwitcherFairlightAudioAuxOutputInput</code>	<code>IID_IBMDSwitcherFairlightAudioAuxOutputInput</code>	An <code>IBMDSwitcherFairlightAudioAuxOutputInputCallback</code> object interface is installed with <code>IBMDSwitcherFairlightAudioAuxOutputInput::AddCallback</code> and removed with <code>IBMDSwitcherFairlightAudioAuxOutputInput::RemoveCallback</code> .

Public Member Functions	
Method	Description
<code>Notify</code>	Called when an event occurs.

7.5.34.1 IBMDSwitcherFairlightAudioAuxOutputInputCallback::Notify method

The `Notify` method is called when `IBMDSwitcherFairlightAudioAuxOutputInput` 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(BMDSwitcherFairlightAudioAuxOutputInputEventType eventType)
```

Parameters

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

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

7.6 Talkback Interface Reference

7.6.1 IBMDSwitcherTalkbackIterator Interface

The `IBMDSwitcherTalkbackIterator` is used to enumerate the available talkbacks.

A reference to an `IBMDSwitcherTalkbackIterator` object interface may be obtained from an `IBMDSwitcher` object interface using the `CreateIterator` method. Pass `IID_IBMDSwitcherTalkbackIterator` for the IID parameter.

Related Interfaces

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

Public Member Functions

Method	Description
Next	Returns a pointer to the next <code>IBMDSwitcherTalkback</code> object interface.
GetById	Returns a pointer to an <code>IBMDSwitcherTalkback</code> object interface, given its <code>BMDSwitcherTalkbackId</code> .

7.6.1.1 IBMDSwitcherTalkbackIterator::Next method

The `Next` method returns the next available `IBMDSwitcherTalkback` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherTalkback* talkback)
```

Parameters

Name	Direction	Description
talkback	out	<code>IBMDSwitcherTalkback</code> object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherTalkback</code> objects available.
E_POINTER	The talkback parameter is invalid.

7.6.1.2 IBMDSwitcherTalkbackIterator::GetById method

The `GetById` method returns a pointer to an `IBMDSwitcherTalkback` object interface, given its `IBMDSwitcherTalkbackId`.

Syntax

```
HRESULT GetById(IBMDSwitcherTalkbackId talkbackId,  
               IBMDSwitcherTalkback* talkback)
```

Parameters

Name	Direction	Description
talkbackId	in	<code>IBMDSwitcherTalkbackId</code> identifier.
talkback	out	<code>IBMDSwitcherTalkback</code> object interface.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_INVALIDARG</code>	The talkbackId is not a valid identifier.
<code>E_POINTER</code>	The talkback parameter is invalid.

7.6.2 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
<code>IBMDSwitcher</code>	<code>IID_IBMDSwitcher</code>	An <code>IBMDSwitcherTalkback</code> interface can be obtained with <code>IBMDSwitcher::QueryInterface</code> .

Public Member Functions	
Method	Description
<code>GetId</code>	Query the talkback ID.
<code>GetMuteSDI</code>	Query the mute state of the talkback SDI input and output channels.
<code>SetMuteSDI</code>	Set the mute state of the talkback SDI input and output channels.
<code>InputCanMuteSDI</code>	Determine if the switcher has the capability of muting the talkback input SDI channels on a particular audio input.
<code>CurrentInputSupportsMuteSDI</code>	Determine if the current external port type of a particular audio input supports muting the talkback input SDI channels.
<code>GetInputMuteSDI</code>	Query the mute state of the talkback SDI channels on a particular audio input.
<code>SetInputMuteSDI</code>	Set the mute state of the talkback SDI channels on a particular audio input.
<code>GetMicrophoneMuted</code>	Query the mute state of the talkback microphone.
<code>SetMicrophoneMuted</code>	Set the mute state of the talkback microphone.
<code>CanMuteMicrophone</code>	Determine if the talkback microphone can be muted.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

7.6.2.1 IBMDSwitcherTalkback::GetId method

The **GetId** method is used to check the ID of the audio talkback channel. The ID will be a **BMDSwitcherTalkbackId** identifier.

Syntax

```
HRESULT GetId(BMDSwitcherTalkbackId* talkbackId)
```

Parameter

Name	Direction	Description
talkbackId	out	Identifier of the talkback object.

Return Values

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

7.6.2.2 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.

7.6.2.3 **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.

7.6.2.4 **IBMDSwitcherTalkback::InputCanMuteSDI** method

The **InputCanMuteSDI** method is used to check if the switcher has the capability to mute the talkback input SDI channels of a particular audio input. The audio input is specified by its **BMDSwitcherAudioInputId** identifier.

Syntax

```
HRESULT InputCanMuteSDI(BMDSwitcherAudioInputId audioInputId, boolean* canMuteSDI)
```

Parameters

Name	Direction	Description
audioInputId	in	BMDSwitcherAudioInputId identifier.
canMuteSDI	out	A Boolean value indicating whether the switcher has the capability to mute the talkback input SDI channels of the specified audio input.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioInputId parameter is not a valid identifier.
E_POINTER	The canMuteSDI parameter is not a valid pointer.

7.6.2.5 **IBMDSwitcherTalkback::CurrentInputSupportsMuteSDI** method

The **CurrentInputSupportsMuteSDI** method is used to check if the current input port of a particular audio input supports muting the talkback input SDI channels. The audio input is identified by its **BMDSwitcherAudioInputId** identifier.

Syntax

```
HRESULT CurrentInputSupportsMuteSDI(BMDSwitcherAudioInputId audioInputId,
boolean* supportsMuteSDI)
```

Parameters

Name	Direction	Description
audioInputId	in	BMDSwitcherAudioInputId identifier.
supportsMuteSDI	out	A Boolean value indicating whether the current audio input port supports muting the talkback SDI channels.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioInputId parameter is not a valid identifier.
E_POINTER	The supportsMuteSDI parameter is not a valid pointer.

7.6.2.6 **IBMDSwitcherTalkback::GetInputMuteSDI** method

The **GetInputMuteSDI** method is used to check the mute state of the audio on the dedicated talkback input SDI channels of a particular audio input. The input is identified by its **BMDSwitcherAudioInputId** identifier.

Syntax

```
HRESULT GetInputMuteSDI(BMDSwitcherAudioInputId audioInputId, boolean* muteSDI)
```

Parameters

Name	Direction	Description
audioInputId	in	BMDSwitcherAudioInputId identifier.
supportsMuteSDI	out	The current mute state of the audio on the talkback input SDI channels of the audio input.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioInputId parameter is not a valid identifier.
E_POINTER	The supportsMuteSDI parameter is not a valid pointer.
E_FAIL	Failure. This can happen if the switcher does not have the capability to mute the talkback input SDI channels of the specified audio input.

7.6.2.7 **IBMDSwitcherTalkback::SetInputMuteSDI** method

The **SetInputMuteSDI** method is used to set the mute state of the audio on the dedicated talkback input SDI channels for a particular audio input. The input is specified by its **BMDSwitcherAudioInputId** identifier.

Syntax

```
HRESULT SetInputMuteSDI(BMDSwitcherAudioInputId audioInputId, boolean muteSDI)
```

Parameters

Name	Direction	Description
audioInputId	in	BMDSwitcherAudioInputId identifier.
muteSDI	in	The desired mute state of the audio on the talkback input SDI channels of the specified audio input.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The audioInputId parameter is not a valid identifier.
E_FAIL	Failure. This can happen if the switcher does not have the capability to mute the talkback input SDI channels of the specified audio input.

7.6.2.8 **IBMDSwitcherTalkback::GetMicrophoneMuted** method

The **GetMuteSDI** method returns the mute state of the talkback microphone.

Syntax

```
HRESULT GetMicrophoneMuted(Boolean* muted)
```

Parameters

Name	Direction	Description
muted	out	The mute state of the talkback microphone.

Return Values

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

7.6.2.9 IBMDSwitcherTalkback::SetMicrophoneMuted method

The `SetMuteSDI` method sets the mute state of the talkback microphone.

Syntax

```
HRESULT SetMicrophoneMuted(Boolean muted)
```

Parameters

Name	Direction	Description
muted	in	The mute state of the talkback microphone.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

7.6.2.10 IBMDSwitcherTalkback::CanMuteMicrophone method

The `CanMuteMicrophone` method is used to determine if the switcher supports muting the talkback microphone.

Syntax

```
HRESULT CanMuteMicrophone(Boolean* canMuteMicrophone)
```

Parameters

Name	Direction	Description
canMuteMicrophone	out	Boolean value indicating whether the talkback microphone can be muted.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The <code>canMuteMicrophone</code> parameter is invalid.
E_FAIL	Failure.

7.6.2.11 **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 IBMDSwitcherTalkbackCallback interface.

Return Values

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

7.6.2.12 **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.

7.6.3 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 IBMDSwitcherTalkbackCallback interface may be installed with IBMDSwitcherTalkback::AddCallback .

Public Member Functions	
Method	Description
Notify	A Switcher talkback event occurred, such as a change in the talkback input SDI mute state.

7.6.3.1 IBMDSwitcherTalkbackCallback::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, BMDSwitcherAudioInputId audioInputId)
```

Parameters

Name	Direction	Description
eventType	in	A BMDSwitcherTalkbackEventType that describes the type of event that has occurred.
audioInputId	in	Specifies the BMDSwitcherAudioInputId of the audio input for the eventType. This parameter is not valid when the eventType is bmdSwitcherTalkbackEventTypeMuteSDIChanged , because that event does not relate to a specific audio input.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 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.

- **bmdSwitcherCameraControlEventTypePeriodicFlushIntervalChanged**
The periodic flush interval has changed.
- **bmdSwitcherCameraControlEventTypeParameterValueChanged**
The parameter value has changed.
- **bmdSwitcherCameraControlEventTypeParameterPeriodicFlushEnabledChanged**
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.

- **bmdSwitcherCameraControlParameterTypeVoidBool**
Boolean (1 or more values) or Void (0 values).
- **bmdSwitcherCameraControlParameterTypeSigned8Bit**
Signed 8-bit type. This type is the same as byte.
- **bmdSwitcherCameraControlParameterTypeSigned16Bit**
Signed 16-bit type
- **bmdSwitcherCameraControlParameterTypeSigned32Bit**
Signed 32-bit type
- **bmdSwitcherCameraControlParameterTypeSigned64Bit**
Signed 64-bit type
- **bmdSwitcherCameraControlParameterTypeFixedPoint16Bit**
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
<code>IBMDSwitcherCameraControl</code>	<code>IID_IBMDSwitcherCameraControl</code>	<code>IBMDSwitcherCameraControl::CreateIterator</code> returns an <code>IBMDSwitcherCameraControlParameterIterator</code> object interface when the <code>IID_IBMDSwitcherCameraControlParameterIterator</code> IID is specified.

Public Member Functions	
Method	Description
<code>Next</code>	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
<code>destinationDevice</code>	out	The destination device address.
<code>category</code>	out	The configuration category number.
<code>parameter</code>	out	The configuration parameter number.

Return Values

Value	Description
<code>S_FALSE</code>	No more control parameters are available.
<code>S_OK</code>	Success.
<code>E_POINTER</code>	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 IBMDSwitcherCameraControlCallback object interface may be installed with IBMDSwitcherCameraControl::AddCallback .

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 IBMDSwitcherCameraControl object interface can be obtained with <code>IBMDSwitcher::QueryInterface</code> .

Public Member Functions	
Method	Description
<code>CreateIterator</code>	Create an iterator.
<code>GetPeriodicFlushInterval</code>	Get the periodic flush interval for parameters to be sent over SDI.
<code>SetPeriodicFlushInterval</code>	Set the periodic flush interval for parameters to be sent over SDI.
<code>GetParameterInfo</code>	Get the type and count of a parameter.
<code>GetParameterPeriodicFlushEnabled</code>	Get the status of flush enable.
<code>SetFlags</code>	Sets the current flag values.
<code>ToggleFlags</code>	Toggles the current flag values.
<code>GetFlags</code>	Gets the current flag values.
<code>SetInt8s</code>	Sets the current signed 8-bit integer values.
<code>OffsetInt8s</code>	Apply signed offsets to the signed 8-bit values.
<code>GetInt8s</code>	Get the current signed 8-bit integer values.
<code>SetInt16s</code>	Sets the current signed 16-bit integer values.
<code>OffsetInt16s</code>	Apply a signed offset to the signed 16-bit integer values.
<code>GetInt16s</code>	Get the current signed 16-bit integer values.
<code>SetInt32s</code>	Sets the current signed 32-bit integer values.
<code>OffsetInt32s</code>	Apply a signed offset to the signed 32-bit integer values.
<code>GetInt32s</code>	Get the current signed 32-bit integer values.
<code>SetInt64s</code>	Sets the current signed 64-bit integer values.
<code>OffsetInt64s</code>	Apply a signed offset to the signed 64-bit integer values.
<code>GetInt64s</code>	Get the current signed 64-bit integer values.
<code>OffsetFloats</code>	Apply a signed floating point offset to the fixed point values.
<code>SetFloats</code>	Sets the current fixed point values.
<code>GetFloats</code>	Get the current fixed point values.
<code>AddCallback</code>	Add a callback to receive camera control event notifications.
<code>RemoveCallback</code>	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 (**IBMDSwitcherCameraControlParameterType**) and the number of values for a given parameter.

Syntax

```
HRESULT GetParameterInfo (uint32_t destinationDevice, uint32_t category,  
uint32_t parameter, IBMDSwitcherCameraControlParameterType* 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
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.

8.2.3.9 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::SetInt8s method

The **SetInt8s** method sends 8-bit values to the cameras over SDI.

Syntax

```
HRESULT SetInt8s (uint32_t destinationDevice, uint32_t category,  
uint32_t parameter, uint32_t count, const int8_t* 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.

8.2.3.11 IBMDSwitcherCameraControl::OffsetInt8s method

The **OffsetInt8s** method will add signed offsets to the current 8-bit values and then send the offset values to the cameras over SDI.

Syntax

```
HRESULT OffsetInt8s (uint32_t destinationDevice, uint32_t category, uint32_t parameter, uint32_t count, const int8_t* 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.

8.2.3.12 IBMDSwitcherCameraControl::GetInt8s method

The **GetInt8s** method returns the last signed bytes sent to the cameras over SDI.

Syntax

```
HRESULT GetInt8s (uint32_t destinationDevice, uint32_t category, uint32_t parameter, uint32_t* count, int8_t* 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
S_OK	Success.
E_INVALIDARG	Values do not exist for the given arguments.
E_POINTER	The count or values 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
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.

8.2.3.14 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
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.

8.2.3.15 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
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 argument is NULL.
E_OUTOFMEMORY	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
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.

8.2.3.18 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
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 argument is NULL.
E_OUTOFMEMORY	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
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.

8.2.3.21 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
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
S_OK	Success.
E_INVALIDARG	Int64s do not exist for the given arguments.
E_POINTER	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
destinationDevice	in	The destination device address.
category	in	The category.
parameter	in	The parameter.
count	in	The element count.
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.

8.2.3.24 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 IBMDSwitcherCameraControlCallback 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 IBMDSwitcherCameraControlCallback object interface.

Return Values

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

Section 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

IBMDSwitcherMacroPoolEventType enumerates the possible event types for the **IBMDSwitcherMacroPoolCallback** object interface.

- **bmdSwitcherMacroPoolEventTypeValidChanged**
A macro has been created (becomes valid), or deleted (becomes invalid).
- **bmdSwitcherMacroPoolEventTypeHasUnsupportedOpsChanged**
A macro's unsupported operations flag has changed.
- **bmdSwitcherMacroPoolEventTypeNameChanged**
A macro's name has changed.
- **bmdSwitcherMacroPoolEventTypeDescriptionChanged**
A macro's description has changed.
- **bmdSwitcherMacroPoolEventTypeTransferCompleted**
A macro transfer has completed.
- **bmdSwitcherMacroPoolEventTypeTransferCancelled**
A macro transfer has cancelled.
- **bmdSwitcherMacroPoolEventTypeTransferFailed**
A macro transfer has failed.

9.2.2 Macro Control Event Type

BMDSwitcherMacroControlEventType enumerates the possible event types for the **IBMDSwitcherMacroControlCallback** object interface.

- **bmdSwitcherMacroControlEventTypeRunStatusChanged**
The switcher's macro playback state has changed.
- **bmdSwitcherMacroControlEventTypeRecordStatusChanged**
The switcher's macro record state has changed.

9.2.3 Macro Run Status

BMDSwitcherMacroRunStatus enumerates the possible macro playback states.

- **bmdSwitcherMacroRunStatusIdle**
No macro is playing.
- **bmdSwitcherMacroRunStatusRunning**
A macro is playing.
- **bmdSwitcherMacroRunStatusWaitingForUser**
A macro is waiting for the user to continue playing.

9.2.4 Macro Record Status

BMDSwitcherMacroRecordStatus enumerates the possible macro record states.

- **bmdSwitcherMacroRecordStatusIdle**
No macro is being recorded.
- **bmdSwitcherMacroRecordStatusRecording**
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 IBMDSwitcherMacroPool object interface can be obtained with the IBMDSwitcher:QueryInterface 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.

Public Member Functions	
Method	Description
GetDescription	Gets a macro's description.
SetDescription	Sets a macro's description.
CreateMacro	Creates an IBMDSwitcherMacro 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 IBMDSwitcherMacro 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	IBMDSwitcherMacro object containing the macro binary data for the transfer.
macroTransfer	out	IBMDSwitcherMacroTransfer 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	IBMDSwitcherMacroTransfer 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 IBMDSwitcherMacroPoolCallback 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 IBMDSwitcherMacroPoolCallback 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	IBMDSwitcherMacroPool::Upload and IBMDSwitcherMacroPool::Download return an IBMDSwitcherTransferMacro object.
IBMDSwitcherMacroPoolCallback	IID_IBMDSwitcherMacroPool	IBMDSwitcherMacroPoolCallback::Notify passes in an IBMDSwitcherTransferMacro object.

Public Member Functions	
Method	Description
Cancel	Cancels the pending transfer.
GetProgress	Gets the pending transfer's progress.
GetMacro	Gets an IBMDSwitcherMacro 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 IBMDSwitcherMacro 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	IBMDSwitcherMacroPool::CreateMacro returns an IBMDSwitcherMacro object.
IBMDSwitcherTransferMacro	IID_IBMDSwitcherMacroTransfer	IBMDSwitcherMacroTransfer::GetMacro returns an IBMDSwitcherMacro 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

```
HRESULT    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 IBMDSwitcherMacroPoolCallback object interface is installed with IBMDSwitcherMacroPool::AddCallback and removed with IBMDSwitcherMacroPool::RemoveCallback

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	BMDSwitcherMacroPoolEventType 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 bmdSwitcherMacroPoolEventTypeTransferCompleted , bmdSwitcherMacroPoolEventTypeTransferCancelled , or bmdSwitcherMacroPoolEventTypeTransferFailed then this parameter is a pointer to the affected IBMDSwitcherTransferMacro 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 IBMDSwitcherMacroControl object interface can be obtained with the IBMDSwitcher::QueryInterface .

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	BMDSwitcherMacroRunStatus 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	BMDSwitcherMacroRecordStatus 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 IBMDSwitcherMacroControlCallback 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 IBMDSwitcherMacroControlCallback 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 IBMDSwitcherMacroControlCallback object interface is installed with IBMDSwitcherMacroControl::AddCallback and removed with IBMDSwitcherMacroControl::RemoveCallback .

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	BMDSwitcherMacroControlEventType that describes the type of event that has occurred.

Return Values

Value	Description
E_FAIL	Failure.
S_OK	Success.

Section 10 — HyperDeck

The Switcher HyperDeck API provides the functionality to remotely control one or more Blackmagic Design HyperDeck devices remotely, to record and playback clips.

10.1 General Information

10.1.1 HyperDeck Interfaces

Different switcher models may support different number of remote HyperDecks. Each HyperDeck interface in the device may connect to a single remote HyperDeck unit, accessible on the same local network.

Commands are issued in an asynchronous manner; they are internally queued and issued to the HyperDeck. As a result, many operations will not cause an immediate change of state until the command has been acknowledged by the HyperDeck.

10.1.2 HyperDeck Remote Control

If a HyperDeck is connected but the remote access mode (set on the HyperDeck itself) is disabled, the interface will effectively be read-only. Commands issued to change the HyperDeck state while remote access is disabled will be rejected.

10.1.3 HyperDeck Clip Cache

To speed up processing, the list of clips located on the active storage media of the HyperDeck is cached locally by the switcher. This cache is automatically invalidated when a recording session ends, or when the active storage media changes, and is signaled by a change in clip count on the HyperDeck's interface. Clients should listen for this change and discard any held clip entry references, re-fetching the new entries from the clip cache when the change occurs. Previously invalidated clip cache entries will remain invalid indefinitely.

The clip cache is filled asynchronously to free up bandwidth for user-initiated actions. This means that cache entries may exist (be valid) but may not have their clip information properties populated until a later point in time.

10.1.4 HyperDeck Configuration

An application for controlling a HyperDeck from the 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 `IBMDSwitcherHyperDeckIterator` object interface
- Obtain an `IBMDSwitcherHyperDeck` object interface using `IBMDSwitcherHyperDeckIterator::Next`
- Use `IBMDSwitcherHyperDeck::SetNetworkAddress` to configure the remote network IPv4 address of a HyperDeck
- Poll `IBMDSwitcherHyperDeck::GetConnectionStatus` to determine when the switcher has a successful connection to a HyperDeck (or listen for changes by installing a callback via `IBMDSwitcherHyperDeck::AddCallback`)

10.1.5 HyperDeck Clip Cache Configuration

An application for reading out the clips in a connected HyperDeck from the switcher may perform the following steps:

- Use `IBMDSwitcherHyperDeck::CreateIterator` to get an `IBMDSwitcherHyperDeckClipIterator` object interface
- Obtain an `IBMDSwitcherHyperDeckClip` object interface using `IBMDSwitcherHyperDeckClipIterator::Next`
- Install a callback via `IBMDSwitcherHyperDeckClip::AddCallback` to listen for information available and invalidation events

10.2 Hyperdeck Types

10.2.1 BMDSwitcherHyperDeckClipEventType

`BMDSwitcherHyperDeckClipEventType` enumerates the possible event types for the `IBMDSwitcherHyperDeckClipCallback` object interface.

- `bmdSwitcherHyperDeckClipEventTypeValidChanged`
A HyperDeck clip has been created (becomes valid) or is no longer available (becomes invalid)
- `bmdSwitcherHyperDeckClipEventTypeInfoAvailableChanged`
A HyperDeck clip's information properties (e.g. name) have become (in-)validated

10.2.2 BMDSwitcherHyperDeckEventType

`BMDSwitcherHyperDeckEventType` enumerates the possible event types for the `IBMDSwitcherHyperDeckCallback` object interface.

- `bmdSwitcherHyperDeckEventTypeConnectionStatusChanged`
The connection status of a HyperDeck to the switcher has changed
- `bmdSwitcherHyperDeckEventTypeRemoteAccessEnabledChanged`
The Remote Access function of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeStorageMediaStateChanged`
The state of a storage media of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeEstimatedRecordTimeRemainingChanged`
The estimated record time available of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeActiveStorageMediaChanged`
The active storage media slot of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeClipCountChanged`
The number of clips available on a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeSwitcherInputChanged`
The switcher input associated with a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeFrameRateChanged`
The video frame rate configuration of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeInterlacedVideoChanged`
The video interlaced configuration of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypeDropFrameTimeCodeChanged`
The video drop frame configuration of a HyperDeck has changed
- `bmdSwitcherHyperDeckEventTypePlayerStateChanged`
The player state of a HyperDeck has changed

- **bmdSwitcherHyperDeckEventTypeCurrentClipChanged**
The currently selected clip of a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeCurrentClipTimeChanged**
The timecode for the currently selected clip in a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeCurrentTimelineTimeChanged**
The timecode for the overall timeline in a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeShuttleSpeedChanged**
The playback speed of a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeLoopedPlaybackChanged**
The playback loop state of a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeSingleClipPlaybackChanged**
The single clip playback mode of a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeAutoRollOnTakeChanged**
The automatic playback of a HyperDeck on Switcher input tally has changed
- **bmdSwitcherHyperDeckEventTypeAutoRollOnTakeFrameDelayChanged**
The frame delay of the automatic playback of a HyperDeck on Switcher input tally has changed
- **bmdSwitcherHyperDeckEventTypeNetworkAddressChanged**
The network address of a HyperDeck has changed
- **bmdSwitcherHyperDeckEventTypeModelChanged**
The model of a HyperDeck has changed.
- **bmdSwitcherHyperDeckEventTypeStorageMediaInfoChanged**
The storage media in a HyperDeck has changed.

10.2.3 **BMDSwitcherHyperDeckClipId**

BMDSwitcherHyperDeckClipId is a signed 64 bit integer type and used as a unique identifier for each clip stored on a media accessible to a HyperDeck.

10.2.4 **BMDSwitcherHyperDeckId**

BMDSwitcherHyperDeckId is a signed 64 bit integer type and used as a unique identifier for each HyperDeck controllable via the attached device.

10.2.5 **BMDSwitcherHyperDeckPlayerState**

BMDSwitcherHyperDeckPlayerState enumerates the possible HyperDeck player states for a HyperDeck.

- **bmdSwitcherHyperDeckStateUnknown**
The state of the attached HyperDeck is not currently known.
- **bmdSwitcherHyperDeckStateIdle**
The HyperDeck is currently idle (stopped).
- **bmdSwitcherHyperDeckStatePlay**
The HyperDeck is currently playing a clip.
- **bmdSwitcherHyperDeckStateRecord**
The HyperDeck is currently recording a clip.
- **bmdSwitcherHyperDeckStateShuttle**
The HyperDeck is currently shuttling.

10.2.6 **BMDSwitcherHyperDeckConnectionStatus**

BMDSwitcherHyperDeckConnectionStatus enumerates the possible HyperDeck connection status states for a **IBMDSwitcherHyperDeck** interface.

- **bmdSwitcherHyperDeckConnectionStatusNotConnected**
A HyperDeck is currently not connected to this interface.
- **bmdSwitcherHyperDeckConnectionStatusConnecting**
This interface is currently attempting to connect to to a HyperDeck.
- **bmdSwitcherHyperDeckConnectionStatusConnected**
A HyperDeck is currently connected to this interface.
- **bmdSwitcherHyperDeckConnectionStatusIncompatible**
A HyperDeck with an incompatible firmware version is currently connected to this interface.

10.2.7 **BMDSwitcherHyperDeckStorageMediaState**

BMDSwitcherHyperDeckStorageMediaState enumerates the possible states of a media slot on an attached HyperDeck.

- **bmdSwitcherHyperDeckStorageMediaStateReady**
This media slot is currently populated and ready for use.
- **bmdSwitcherHyperDeckStorageMediaStateUnavailable**
This media slot is currently empty or cannot be accessed.

10.2.8 **BMDSwitcherHyperDeckErrorType**

BMDSwitcherHyperDeckErrorType enumerates the possible transient errors that may occurs on attached HyperDeck.

- **bmdSwitcherHyperDeckErrorTypeUnknown**
An unknown error has occurred on the HyperDeck.
- **bmdSwitcherHyperDeckErrorTypeAlreadyInUse**
Another client is currently remotely accessing the HyperDeck.
- **bmdSwitcherHyperDeckErrorTypeRemoteDisabled**
A request was rejected by the Hyperdeck (remote access is disabled).
- **bmdSwitcherHyperDeckErrorTypeMediaFull**
The storage media of the HyperDeck is full and new clips could not be added.
- **bmdSwitcherHyperDeckErrorTypeMediaError**
An error occurred while attempting to access the storage media of the HyperDeck.
- **bmdSwitcherHyperDeckErrorTypeNoInput**
There is no video signal detected on the HyperDeck input.
- **bmdSwitcherHyperDeckErrorTypeDuplicateAddress**
Two or more HyperDeck interfaces have been configured with the same IP address.

10.3 Interface Reference

10.3.1 IBMDSwitcherHyperDeckIterator Interface

The `IBMDSwitcherHyperDeckIterator` is used to enumerate the available HyperDecks.

Related Interfaces

Interface	Interface ID	Description
BMDSwitcher	IID_IBMDSwitcher	<code>IBMDSwitcher::CreateIterator</code> can return an <code>IBMDSwitcherHyperDeckIterator</code> object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next <code>IBMDSwitcherHyperDeck</code> object interface.
GetById	Returns a specific <code>IBMDSwitcherHyperDeck</code> object by its ID.

10.3.1.1 IBMDSwitcherHyperDeckIterator::Next method

The Next method returns the next available `IBMDSwitcherHyperDeck` object interface.

Syntax

```
HRESULT Next (IBMDSwitcherHyperDeck* hyperDeck);
```

Parameters

Name	Direction	Description
hyperDeck	out	<code>IBMDSwitcherHyperDeck</code> object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherHyperDeck</code> objects available.
E_POINTER	The hyperDeck parameter is invalid.

10.3.1.2 IBMDSwitcherHyperDeckIterator::GetById method

The `GetById` method returns the `IBMDSwitcherHyperDeck` object interface that matches the given ID.

Syntax

```
HRESULT GetById (IBMDSwitcherHyperDeckId hyperDeckId, IBMDSwitcherHyperDeck* hyperDeck);
```

Parameters

Name	Direction	Description
hyperDeckId	in	IBMDSwitcherHyperDeckId to retrieve.
hyperDeck	out	IBMDSwitcherHyperDeck object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The requested HyperDeck was not found.
E_POINTER	The hyperDeck parameter is invalid.

10.3.2 IBMDSwitcherHyperDeck Interface

The `IBMDSwitcherHyperDeck` object interface provides functionality for the remote control of a Blackmagic Design HyperDeck device.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherHyperDeckIterator	IID_IBMDSwitcherHyperDeckIterator	An <code>IBMDSwitcherHyperDeck</code> object will be returned after a successful call to <code>IBMDSwitcherHyperDeckIterator::Next</code> method.

Public Member Functions	
Method	Description
GetId	Returns the ID of this <code>IBMDSwitcherHyperDeck</code> interface.
GetConnectionStatus	Returns the current connection status of the interface to a remote HyperDeck device.
IsRemoteAccessEnabled	Returns whether remote access is enabled.
GetStorageMediaCount	Returns the number of media slots supported by the remote HyperDeck.
GetStorageMediaState	Returns the state of a media slot in a remote HyperDeck.
GetActiveStorageMedia	Returns the index of the currently active media slot in the remote HyperDeck.
SetActiveStorageMedia	Selects the currently active media slot in the remote HyperDeck.
GetClipCount	Returns the number of clips in the active media in the remote HyperDeck.
CreateIterator	Create an iterator.
GetSwitcherInput	Returns the currently associated Switcher input.
SetSwitcherInput	Sets the associated Switcher input.

Public Member Functions	
Method	Description
GetFrameRate	Returns the configured video frame rate of the HyperDeck.
IsInterlacedVideo	Returns whether the configured video mode is an interlaced mode.
IsDropFrameTimeCode	Returns whether the configured video mode supports drop frames.
GetPlayerState	Returns the current player state.
GetCurrentClip	Returns the currently selected clip ID in the active storage slot.
SetCurrentClip	Sets the currently selected clip ID in the active storage slot.
GetCurrentClipTime	Returns the current timecode time within the currently selected clip.
SetCurrentClipTime	Sets the current timecode within the currently selected clip.
GetCurrentTimelineTime	Returns the current timeline timecode.
SetCurrentTimelineTime	Sets the current timeline timecode.
GetEstimatedRecordTimeRemaining	Returns the current estimated recording time across all slots.
Play	Starts playback with the current settings.
Record	Starts a new clip recording session.
Stop	Stops playback.
Shuttle	Shuttles at the given playback speed.
GetShuttleSpeed	Returns the current shuttle speed.
Jog	Jogs one or more frames in the timeline.
GetLoopedPlayback	Returns whether playback requests will loop when the clip(s) have finished playing.
SetLoopedPlayback	Sets whether playback requests will loop when the clip(s) have finished playing.
GetSingleClipPlayback	Returns whether playback requests will be bound to the currently selected clip, or the entire timeline.
SetSingleClipPlayback	Sets whether playback requests will be bound to the currently selected clip, or the entire timeline.
GetAutoRollOnTake	Sets whether the HyperDeck automatically rolls when the HyperDeck is tallied to program output.
SetAutoRollOnTake	Returns whether the HyperDeck automatically rolls when the HyperDeck is tallied to program output.
GetAutoRollOnTakeFrameDelay	Returns the automatic roll-on-take frame delay when the HyperDeck is tallied.
SetAutoRollOnTakeFrameDelay	Sets the automatic roll-on-take frame delay when the HyperDeck is tallied.
GetNetworkAddress	Returns the currently set network address of a HyperDeck interface.
SetNetworkAddress	Sets the network address of a HyperDeck interface.
GetModelName	Returns the model name of the Hyperdeck.
GetStorageMediaInfo	Returns information about storage media on the Hyperdeck.
AddCallback	Add a HyperDeck callback to receive property changes.
RemoveCallback	Remove a HyperDeck callback.

10.3.2.1 IBMDSwitcherHyperDeck::GetId method

The **GetId** method returns the HyperDeck's ID, used to uniquely identify a HyperDeck control interface within the Switcher.

Syntax

```
HRESULT GetId (BMDSwitcherHyperDeckId* hyperDeckId);
```

Parameters

Name	Direction	Description
hyperDeckId	out	BMDSwitcherHyperDeckId identifier for the current interface.

Return Values

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

10.3.2.2 IBMDSwitcherHyperDeck::GetConnectionStatus method

The **GetConnectionStatus** method returns the HyperDeck's connection status, used to determine the connection state of a HyperDeck control interface within the Switcher.

Syntax

```
HRESULT GetConnectionStatus (BMDSwitcherHyperDeckConnectionStatus* status);
```

Parameters

Name	Direction	Description
status	out	BMDSwitcherHyperDeckConnectionStatus connection state for the interface.

Return Values

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

10.3.2.3 IBMDSwitcherHyperDeck::IsRemoteAccessEnabled method

The **IsRemoteAccessEnabled** method returns the HyperDeck's remote access status. A connected HyperDeck is only remotely controllable if the remote access function of the HyperDeck is enabled.

A connected HyperDeck with remote access disabled is effectively read-only; set requests will be rejected by the HyperDeck and will generate a **NotifyError** callback (if a handler is installed).

Syntax

```
HRESULT IsRemoteAccessEnabled (boolean* enabled);
```

Parameters

Name	Direction	Description
enabled	out	Boolean remote access enable state of the remote HyperDeck.

Return Values

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

10.3.2.4 IBMDSwitcherHyperDeck::GetStorageMediaCount method

The **GetStorageMediaCount** method returns the number of physical media storage slots on the connected HyperDeck device.

Syntax

```
HRESULT GetStorageMediaCount (uint32_t* count);
```

Parameters

Name	Direction	Description
count	out	Number of media slots on the connected HyperDeck.

Return Values

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

10.3.2.5 IBMDSwitcherHyperDeck::GetStorageMediaState method

The **GetStorageMediaState** method returns the state of a given storage media slot on a connected HyperDeck.

Syntax

```
HRESULT GetStorageMediaState (uint32_t storageMediaId,  
                              BMDSwitcherHyperDeckStorageMediaState* state);
```

Parameters

Name	Direction	Description
storageMediaId	in	Storage media slot to examine.
state	out	Current state of the storage media slot.

Return Values

Value	Description
S_OK	Success.
S_FALSE	Given slot index was out of bounds.
E_FAIL	Failure.
E_POINTER	The state parameter is invalid.

10.3.2.6 IBMDSwitcherHyperDeck::GetActiveStorageMedia method

The **GetActiveStorageMedia** method returns the index of the active storage media slot on a connected HyperDeck.

If no storage media is active, this will return a -1 slot index.

Syntax

```
HRESULT GetActiveStorageMedia (uint32_t* index);
```

Parameters

Name	Direction	Description
index	out	Active slot index of the connected HyperDeck.

Return Values

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

10.3.2.7 IBMDSwitcherHyperDeck::SetActiveStorageMedia method

The `SetActiveStorageMedia` method sets the active storage media slot on a connected HyperDeck.

Syntax

```
HRESULT SetActiveStorageMedia (uint32_t index);
```

Parameters

Name	Direction	Description
index	in	Active slot index to select in the connected HyperDeck.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.8 IBMDSwitcherHyperDeck::GetClipCount method

The `GetClipCount` method returns the total number of clips in the active storage media slot on a connected HyperDeck.

Syntax

```
HRESULT GetClipCount (uint32_t* count);
```

Parameters

Name	Direction	Description
count	out	Number of clips in the active slot of the connected HyperDeck.

Return Values

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

10.3.2.9 **IBMDSwitcherHyperDeck::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	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.
E_NOINTERFACE	Interface was not found.

10.3.2.10 **IBMDSwitcherHyperDeck::GetSwitcherInput** method

The **GetSwitcherInput** method returns the associated Switcher input source of the HyperDeck interface. This is used to detect when a HyperDeck has tally, in particular for the roll-on-take feature.

An associated input of zero is used as a sentinel for no associated input.

Syntax

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

Parameters

Name	Direction	Description
inputId	out	Associated Switcher input ID of the HyperDeck interface.

Return Values

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

10.3.2.11 **IBMDSwitcherHyperDeck::SetSwitcherInput** method

The **SetSwitcherInput** method sets the associated Switcher input source of the HyperDeck interface. This is used to detect when a HyperDeck has tally, in particular for the roll-on-take feature.

An associated input of zero is used as a sentinel for no associated input. Only external input sources are valid source inputs; an internal source will be converted to zero.

Syntax

```
HRESULT SetSwitcherInput (BMDSwitcherInputId inputId);
```

Parameters

Name	Direction	Description
inputId	in	Switcher input ID to be associated with the HyperDeck interface.

Return Values

Value	Description
S_OK	Success.

10.3.2.12 **IBMDSwitcherHyperDeck::GetFrameRate** method

The **GetFrameRate** method retrieves the configured video frame rate of the connected HyperDeck.

The framerate is expressed as a number of frames per timescale units, i.e. frames per second is frameRate divided by timescale.

Syntax

```
HRESULT GetFrameRate (uint32_t* frameRate, uint32_t* timeScale);
```

Parameters

Name	Direction	Description
frameRate	out	Video frame rate (numerator) of the configured HyperDeck.
timeScale	out	Timescale (denominator) of the video frame rate.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The frameRate or timeScale parameter(s) are invalid.

10.3.2.13 **IBMDSwitcherHyperDeck::IsInterlacedVideo** method

The **IsInterlacedVideo** returns whether the configured video mode is an interlaced mode.

Syntax

```
HRESULT IsInterlacedVideo (boolean* isInterlaced);
```

Parameters

Name	Direction	Description
isInterlaced	out	Boolean value of the interlaced video setting of the configured HyperDeck.

Return Values

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

10.3.2.14 **IBMDSwitcherHyperDeck::IsDropFrameTimeCode** method

The **IsDropFrameTimeCode** method retrieves the configured video drop frame timecode setting of the connected HyperDeck.

Syntax

```
HRESULT IsDropFrameTimeCode (boolean* isDropFrame);
```

Parameters

Name	Direction	Description
isDropFrame	out	Boolean value of the drop frame timecode video setting of the configured HyperDeck.

Return Values

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

10.3.2.15 IBMDSwitcherHyperDeck::GetPlayerState method

The `GetPlayerState` method retrieves the current player state of the connected HyperDeck.

Syntax

```
HRESULT GetPlayerState (BMDSwitcherHyperDeckPlayerState* playerState);
```

Parameters

Name	Direction	Description
playerState	out	Current player state of the connected HyperDeck.

Return Values

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

10.3.2.16 IBMDSwitcherHyperDeck::GetCurrentClip method

The `GetCurrentClip` method retrieves the currently selected clip in the active storage media of the connected HyperDeck.

If no clip is selected, this will return a -1 clip ID.

Syntax

```
HRESULT GetCurrentClip (BMDSwitcherHyperDeckClipId* clipId);
```

Parameters

Name	Direction	Description
clipId	out	Currently selected clip ID of the connected HyperDeck.

Return Values

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

10.3.2.17 **IBMDSwitcherHyperDeck::SetCurrentClip** method

The **SetCurrentClip** method sets the currently selected clip in the active storage media of the connected HyperDeck.

Syntax

```
HRESULT SetCurrentClip (BMDSwitcherHyperDeckClipId clipId);
```

Parameters

Name	Direction	Description
clipId	in	Clip ID to select in the connected HyperDeck.

Return Values

Value	Description
S_OK	Success.

10.3.2.18 **IBMDSwitcherHyperDeck::GetCurrentClipTime** method

The **GetCurrentClipTime** method gets the currently selected clip's elapsed time.

Syntax

```
HRESULT GetCurrentClipTime (uint16_t* hours, uint8_t* minutes,  
uint8_t* seconds, uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Elapsed clip time, hours component.
minutes	out	Elapsed clip time, minutes component.
seconds	out	Elapsed clip time, seconds component.
frames	out	Elapsed clip time, frames component.

Return Values

Value	Description
S_OK	Success.
S_FALSE	A clip is not currently selected.
E_POINTER	The hours, minutes, seconds and/or frames parameter is invalid.

10.3.2.19 IBMDSwitcherHyperDeck::SetCurrentClipTime method

The `SetCurrentClipTime` method seeks the currently selected clip to the specified elapsed time.

Syntax

```
HRESULT SetCurrentClipTime (uint16_t hours, uint8_t minutes,
                           uint8_t seconds, uint8_t frames);
```

Parameters

Name	Direction	Description
hours	in	Elapsed clip time, hours component.
minutes	in	Elapsed clip time, minutes component.
seconds	in	Elapsed clip time, seconds component.
frames	in	Elapsed clip time, frames component.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The given position does not exist within the currently selected clip.
E_FAIL	Could not compute a valid timecode for the specified position.

10.3.2.20 IBMDSwitcherHyperDeck::GetCurrentTimelineTime method

The `GetCurrentTimelineTime` method retrieves the current elapsed time within the Hyperdeck's entire timeline.

Syntax

```
HRESULT GetCurrentTimelineTime (uint16_t* hours,
                               uint8_t* minutes, uint8_t* seconds, uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Elapsed timeline time, hours component.
minutes	out	Elapsed timeline time, minutes component.
seconds	out	Elapsed timeline time, seconds component.
frames	out	Elapsed timeline time, frames component.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The hours, minutes, seconds and/or frames parameter is invalid.

10.3.2.21 IBMDSwitcherHyperDeck::SetCurrentTimelineTime method

The **SetCurrentTimelineTime** method sets the current elapsed time within the Hyperdeck's entire timeline.

Syntax

```
HRESULT SetCurrentTimelineTime (uint16_t hours, uint8_t minutes,  
                                uint8_t seconds, uint8_t frames);
```

Parameters

Name	Direction	Description
hours	in	Elapsed timeline time, hours component.
minutes	in	Elapsed timeline time, minutes component.
seconds	in	Elapsed timeline time, seconds component.
frames	in	Elapsed timeline time, frames component.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Could not compute a valid timecode for the specified position.

10.3.2.22 IBMDSwitcherHyperDeck::GetEstimatedRecordTimeRemaining method

The **GetEstimatedRecordTimeRemaining** method gets the estimated recording time remaining across the available media slots, based on the currently configured video mode settings. Note that due to compression, this value may change in a non-linear fashion.

Syntax

```
HRESULT GetEstimatedRecordTimeRemaining (uint16_t* hours, uint8_t* minutes,  
                                          uint8_t* seconds, uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Estimated remaining record time, hours component.
minutes	out	Estimated remaining record time, minutes component.
seconds	out	Estimated remaining record time, seconds component.
frames	out	Estimated remaining record time, frames component.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The hours, minutes, seconds and/or frames parameter is invalid.

10.3.2.23 IBMDSwitcherHyperDeck::Play method

The **Play** method starts playing the currently selected clip on the connected HyperDeck at the current timeline position.

Syntax

```
HRESULT Play (void);
```

Parameters

None.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.24 IBMDSwitcherHyperDeck::Record method

The **Record** method starts recording on the connected HyperDeck at the current timeline position.

Syntax

```
HRESULT Record (void);
```

Parameters

None.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.25 IBMDSwitcherHyperDeck::Stop method

The **Stop** method stops playback on the connected HyperDeck.

Syntax

```
HRESULT Stop (void);
```

Parameters

None.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.26 IBMDSwitcherHyperDeck::Shuttle method

The **Shuttle** method starts playback on the connected HyperDeck at the requested speed.

Syntax

```
HRESULT Shuttle (int32_t speedPercent);
```

Parameters

Name	Direction	Description
speedPercent	in	Shuttle speed, expressed as a percentage.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.27 IBMDSwitcherHyperDeck::GetShuttleSpeed method

The **GetShuttleSpeed** method retrieves the current shuttle playback speed on the connected HyperDeck.

Syntax

```
HRESULT GetShuttleSpeed (int32_t* speedPercent);
```

Parameters

Name	Direction	Description
speedPercent	out	Shuttle speed, expressed as a percentage.

Return Values

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

10.3.2.28 IBMDSwitcherHyperDeck::Jog method

The **Jog** method moves the timeline position forwards or backwards by the specified number of frames.

Syntax

```
HRESULT Jog (int32_t frameDelta);
```

Parameters

Name	Direction	Description
frameDelta	in	Number of frames to jog.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.29 IBMDSwitcherHyperDeck::GetLoopedPlayback method

The `GetLoopedPlayback` method retrieves the current loop state of the connected HyperDeck.

Syntax

```
HRESULT GetLoopedPlayback (boolean* loop);
```

Parameters

Name	Direction	Description
loop	out	Current loop state.

Return Values

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

10.3.2.30 IBMDSwitcherHyperDeck::SetLoopedPlayback method

The `SetLoopedPlayback` method sets the current loop state of the connected HyperDeck.

Syntax

```
HRESULT SetLoopedPlayback (boolean loop);
```

Parameters

Name	Direction	Description
loop	in	New loop state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.31 IBMDSwitcherHyperDeck::GetSingleClipPlayback method

The `GetSingleClipPlayback` method retrieves the current single clip playback state of the connected HyperDeck.

Syntax

```
HRESULT GetSingleClipPlayback (boolean* single);
```

Parameters

Name	Direction	Description
single	out	Current single clip playback state.

Return Values

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

10.3.2.32 IBMDSwitcherHyperDeck::SetSingleClipPlayback method

The `SetSingleClipPlayback` method sets the single clip playback state of the connected HyperDeck.

Syntax

```
HRESULT SetSingleClipPlayback (boolean single);
```

Parameters

Name	Direction	Description
single	in	New single clip state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.33 **IBMDSwitcherHyperDeck::GetAutoRollOnTake** method

The **GetAutoRollOnTake** method retrieves the current roll-on-take (automatic playback on tally) state of the HyperDeck interface.

Syntax

```
HRESULT GetAutoRollOnTake (boolean* autoRollOnTake);
```

Parameters

Name	Direction	Description
autoRollOnTake	out	Current automatic roll on take state.

Return Values

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

10.3.2.34 **IBMDSwitcherHyperDeck::SetAutoRollOnTake** method

The **SetAutoRollOnTake** method sets the current roll-on-take (automatic playback on tally) state of the HyperDeck interface.

If this feature is enabled, and the input associated with this HyperDeck interface (set by **SetSwitcherInput**) is tallied to the program output, the HyperDeck will automatically begin playing at the current timeline position after the frame delay set by **SetAutoRollOnTakeFrameDelay**.

Syntax

```
HRESULT SetAutoRollOnTake (boolean autoRollOnTake);
```

Parameters

Name	Direction	Description
autoRollOnTake	in	New automatic roll on take state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.35 IBMDSwitcherHyperDeck::GetAutoRollOnTakeFrameDelay method

The `GetAutoRollOnTakeFrameDelay` method retrieves the current automatic playback on tally state frame delay of the HyperDeck interface.

Syntax

```
HRESULT GetAutoRollOnTakeFrameDelay (uint16_t* frameDelay);
```

Parameters

Name	Direction	Description
frameDelay	out	Current automatic roll on take frame delay.

Return Values

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

10.3.2.36 IBMDSwitcherHyperDeck::SetAutoRollOnTakeFrameDelay method

The `SetAutoRollOnTakeFrameDelay` method sets the automatic playback on tally state frame delay of the HyperDeck interface.

Syntax

```
HRESULT SetAutoRollOnTakeFrameDelay (uint16_t frameDelay);
```

Parameters

Name	Direction	Description
autoRollOnTake	in	New automatic roll on take frame delay.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.37 IBMDSwitcherHyperDeck::GetNetworkAddress method

The **GetNetworkAddress** method retrieves the current remote device network address of the HyperDeck interface. This is expressed as a packed IPv4 address, least significant byte first.

Syntax

```
HRESULT GetNetworkAddress (uint32_t* networkAddress);
```

Parameters

Name	Direction	Description
networkAddress	out	Currently set remote network IP address of the device.

Return Values

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

10.3.2.38 IBMDSwitcherHyperDeck::SetNetworkAddress method

The **SetNetworkAddress** method sets the remote device network address of the HyperDeck interface. This is expressed as a packed IPv4 address, least significant byte first.

The switcher will continuously attempt to connect to the device at the configured network address. To disable the HyperDeck interface and suppress connection attempts from the switcher, set this to an address of zero.

Syntax

```
HRESULT SetNetworkAddress (uint32_t networkAddress);
```

Parameters

Name	Direction	Description
networkAddress	in	New remote network IP address of the device.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.2.39 IBMDSwitcherHyperDeck::GetModelName method

The `GetModelName` method returns the model name of the connected HyperDeck..

Syntax

```
HRESULT GetModelName(string* model)
```

Parameters

Name	Direction	Description
model	out	Model name of the connected Hyperdeck.

Return Values

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

10.3.2.40 IBMDSwitcherHyperDeck::GetStorageMediaInfo method

The `GetStorageMediaInfo` method returns information for a storage media slot on the connected Hyperdeck.

Syntax

```
HRESULT GetStorageMediaInfo(int16_t slotIndex, string* slotName, string* volumeName)
```

Parameters

Name	Direction	Description
slotIndex	in	Storage media slot to examine.
slotName	out	The name of the slot, indicating its type.
volumeName	out	The volume name of the media in the slot.

10.3.2.41 **IBMDSwitcherHyperDeck::AddCallback** method

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

Pass an object implementing the **IBMDSwitcherHyperDeckCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the in IBMDSwitcherHyperDeckCallback object interface.

Return Values

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

10.3.2.42 **IBMDSwitcherHyperDeck::RemoveCallback** method

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

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherHyperDeckCallback object interface.

Return Values

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

10.3.3 IBMDSwitcherHyperDeckCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
BMDSwitcherHyperDeck	IID_IBMDSwitcherHyperDeck	An IBMDSwitcherHyperDeckCallback object interface is installed with IBMDSwitcherHyperDeck::AddCallback and removed with IBMDSwitcherHyperDeck::RemoveCallback

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

10.3.3.1 IBMDSwitcherHyperDeckCallback::Notify method

The **Notify** method is called when **IBMDSwitcherHyperDeck** 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 (BMDSwitcherHyperDeckEventType eventType);
```

Parameters

Name	Direction	Description
eventType	out	BMDSwitcherHyperDeckEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.3.2 IBMDSwitcherHyperDeckCallback::NotifyError method

The NotifyError method is called when **IBMDSwitcherHyperDeck** error 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 NotifyError (BMDSwitcherHyperDeckErrorType errorType);
```

Parameters

Name	Direction	Description
errorType	out	BMDSwitcherHyperDeckErrorType that describes the type of error that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

10.3.4 IBMDSwitcherHyperDeckClipIterator Interface

The **IBMDSwitcherHyperDeckClipIterator** is used to enumerate the available clips in a connected HyperDeck.

Related Interfaces

Interface	Interface ID	Description
BMDSwitcherHyperDeck	IID_IBMDSwitcherHyperDeck	IBMDSwitcherHyperDeck::CreateIterator can return an IBMDSwitcherHyperDeckClipIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherHyperDeckClip object interface.
GetById	Returns a specific IBMDSwitcherHyperDeckClip object by its clip ID.

10.3.4.1 IBMDSwitcherHyperDeckClipIterator::Next method

The `Next` method returns the next available `IBMDSwitcherHyperDeckClip` object interface.

Syntax

```
HRESULT Next (IBMDSwitcherHyperDeckClip** clip);
```

Parameters

Name	Direction	Description
clip	out	IBMDSwitcherHyperDeckClip object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherHyperDeckClip</code> objects available.
E_POINTER	The clip parameter is invalid.

10.3.4.2 IBMDSwitcherHyperDeckClipIterator::GetById method

The `GetById` method returns the `IBMDSwitcherHyperDeckClip` object interface that matches the given clip ID.

Syntax

```
HRESULT GetById (BMDSwitcherHyperDeckClipId clipId, IBMDSwitcherHyperDeckClip** clip);
```

Parameters

Name	Direction	Description
clipId	in	BMDSwitcherHyperDeckClipId clip to retrieve.
clip	out	IBMDSwitcherHyperDeckClip object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The requested clip was not found.
E_POINTER	The clip parameter is invalid.

10.3.5 IBMDSwitcherHyperDeckClip Interface

The **IBMDSwitcherHyperDeckClip** object interface provides information about a single clip in a Blackmagic Design HyperDeck device.

Related Interfaces

Interface	Interface ID	Description
BMDSwitcherHyperDeckClipIterator	IID_IBMDSwitcherHyperDeckClipIterator	An IBMDSwitcherHyperDeckClip object will be returned after a successful call to IBMDSwitcherHyperDeckClipIterator::Next method.

Public Member Functions	
Method	Description
IsValid	Returns whether this clip entry is still valid, or has been permanently invalidated.
IsInfoAvailable	Returns whether the clip data has been successfully retrieved at this stage.
GetId	Get the ID of the clip.
GetName	Get the name of the clip.
GetDuration	Get the duration timecode of the clip.
GetTimelineStart	Get the start of the clip timecode within the Hyperdeck's timeline.
GetTimelineEnd	Get the end of the clip timecode within the Hyperdeck's timeline.
AddCallback	Add a HyperDeck clip callback to receive property changes.
RemoveCallback	Remove a HyperDeck clip callback.

10.3.5.1 IBMDSwitcherHyperDeckClip::IsValid method

The **IsValid** method returns whether the clip entry has been permanently invalidated. Under certain circumstances (such as a media slot state change) the clip cache becomes invalidated, and all entries are recreated and repopulated from the HyperDeck.

Once a clip has become invalid it should be released and discarded, and a fresh version requested from the **IBMDSwitcherHyperDeckClipIterator** object.

Syntax

```
HRESULT IsValid (boolean* isValid);
```

Parameters

Name	Direction	Description
isValid	out	Boolean value indicating if the clip's information is currently available.

Return Values

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

10.3.5.2 **IBMDSwitcherHyperDeckClip::IsInfoAvailable** method

The **IsInfoAvailable** method returns whether the information for this is currently populated. Clip information is retrieved asynchronously, thus clip entries may be created before their associated data becomes available.

Clips whose information is not currently populated may still have user callbacks registered via **AddCallback**.

Syntax

```
HRESULT IsInfoAvailable (boolean* infoAvailable);
```

Parameters

Name	Direction	Description
infoAvailable	out	Boolean value indicating if the clip's information is current available.

Return Values

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

10.3.5.3 **IBMDSwitcherHyperDeckClip::GetId** method

The **GetId** method returns the clip's ID, used to uniquely identify a clip in the clip cache.

Syntax

```
HRESULT GetId (BMDSwitcherHyperDeckClipId* clipId);
```

Parameters

Name	Direction	Description
clipId	out	BMDSwitcherHyperDeckClipId clip identifier in the cache.

Return Values

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

10.3.5.4 IBMDSwitcherHyperDeckClip::GetName method

The `GetName` method returns the clip's name, as returned by the HyperDeck.

Syntax

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

Parameters

Name	Direction	Description
name	out	Clip name in the cache.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The clip name is not yet available.
E_FAIL	The clip is invalid.
E_POINTER	The name parameter is invalid.

10.3.5.5 IBMDSwitcherHyperDeckClip::GetDuration method

The `GetDuration` method returns the clip's duration, as returned by the HyperDeck.

Syntax

```
HRESULT GetDuration (uint16_t* hours, uint8_t* minutes, uint8_t* seconds,  
uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Clip duration, hours component.
minutes	out	Clip duration, minutes component.
seconds	out	Clip duration, seconds component.
frames	out	Clip duration, frames component.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The clip duration is not yet available.
E_FAIL	The clip is invalid.
E_POINTER	The hours, minutes, seconds and/or frames parameter(s) are invalid.

10.3.5.6 IBMDSwitcherHyperDeckClip::GetTimelineStart method

The `GetTimelineStart` method returns the clip's start position within the entire timeline, as returned by the HyperDeck.

Syntax

```
HRESULT GetTimelineStart (uint16_t* hours, uint8_t* minutes,  
                          uint8_t* seconds, uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Clip timeline start, hours component.
minutes	out	Clip timeline start, minutes component.
seconds	out	Clip timeline start, seconds component.
frames	out	Clip timeline start, frames component.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The clip timeline start is not yet available.
E_FAIL	The clip is invalid.
E_POINTER	The hours, minutes, seconds and/or frames parameter(s) are invalid.

10.3.5.7 IBMDSwitcherHyperDeckClip::GetTimelineEnd method

The `GetTimelineEnd` method returns the clip's end position within the entire timeline, as returned by the HyperDeck.

Syntax

```
HRESULT GetTimelineEnd (uint16_t* hours, uint8_t* minutes,  
                       uint8_t* seconds, uint8_t* frames);
```

Parameters

Name	Direction	Description
hours	out	Clip timeline end, hours component.
minutes	out	Clip timeline end, minutes component.
seconds	out	Clip timeline end, seconds component.
frames	out	Clip timeline end, frames component.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The clip timeline start is not yet available.
E_FAIL	The clip is invalid.
E_POINTER	The hours, minutes, seconds and/or frames parameter(s) are invalid.

10.3.5.8 IBMDSwitcherHyperDeckClip::AddCallback method

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

Pass an object implementing the **IBMDSwitcherHyperDeckClipCallback** interface to receive callbacks. Adding a new callback will not affect previously added callbacks.

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherHyperDeckClipCallback object interface.

Return Values

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

10.3.5.9 IBMDSwitcherHyperDeckClip::RemoveCallback method

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

Syntax

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

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherHyperDeckClipCallback object interface.

Return Values

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

10.3.6 IBMDSwitcherHyperDeckClipCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
BMDSwitcherHyperDeckClip	IID_IBMDSwitcherHyperDeckClip	An IBMDSwitcherHyperDeckClipCallback object interface is installed with IBMDSwitcherHyperDeckClip::AddCallback and removed with IBMDSwitcherHyperDeckClip::RemoveCallback

Public Member Functions

Method	Description
Notify	Called when an event occurs.

10.3.6.1 IBMDSwitcherHyperDeckClipCallback::Notify method

The **Notify** method is called when **IBMDSwitcherHyperDeckClip** 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 (BMDSwitcherHyperDeckClipEventType eventType,  
                BMDSwitcherHyperDeckClipId clipId);
```

Parameters

Name	Direction	Description
eventType	out	BMDSwitcherHyperDeckEventType that describes the type of event that has occurred.
clipId	out	BMDSwitcherHyperDeckClipId identifier of the clip generating the notification.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 11 — Streaming

The Switcher Streaming API provides functionality to stream video over the internet to a streaming service such as YouTube, Twitch, or Facebook.

11.1 General Information

11.1.1 Video and Audio Encoding

The switcher will stream H.264 or H.265 encoded video and AAC encoded audio via the RTMP or SRT streaming protocols to a selected streaming server. This requires that the switcher has access to an internet connection from its ethernet port.

11.2 Streaming Data Types

11.2.1 Streaming State

BMDSwitcherStreamRTMPState enumerates the possible streaming states.

- **bmdSwitcherStreamRTMPStateIdle**
Not streaming.
- **bmdSwitcherStreamRTMPStateConnecting**
Connecting to the streaming server.
- **bmdSwitcherStreamRTMPStateStreaming**
Streaming is in progress.
- **bmdSwitcherStreamRTMPStateStopping**
Streaming is stopping.

11.2.2 Streaming Error

BMDSwitcherStreamRTMPErrors enumerates the possible errors that can occur when streaming.

- **bmdSwitcherStreamRTMPErrorsNone**
No error.
- **bmdSwitcherStreamRTMPErrorsInvalidState**
Invalid state for requested command.
- **bmdSwitcherStreamRTMPErrorsUnknown**
Unknown error.

11.2.3 Streaming Event Type

BMDSwitcherStreamRTMPEventType enumerates the possible event types for the **IBMDSwitcherStreamRTMPCallback** object interface.

- **bmdSwitcherStreamRTMPEventTypeServiceNameChanged**
The service name has changed.
- **bmdSwitcherStreamRTMPEventTypeUrlChanged**
The server url has changed.
- **bmdSwitcherStreamRTMPEventTypeKeyChanged**
The stream key has changed.
- **bmdSwitcherStreamRTMPEventTypeVideoBitratesChanged**
The maximum video bitrates have changed.
- **bmdSwitcherStreamRTMPEventTypeAudioBitratesChanged**
The maximum audio bitrates have changed.
- **bmdSwitcherStreamRTMPEventTypeDownConvertChanged**
The down convert mode has changed.
- **bmdSwitcherStreamRTMPEventTypeEncodingBitrateChanged**
The current encoding bitrate has changed.
- **bmdSwitcherStreamRTMPEventTypeCacheUsedChanged**
The current usage level of the streaming cache has changed.
- **bmdSwitcherStreamRTMPEventTypeTimecodeChanged**
The current streaming timecode has changed.
- **bmdSwitcherStreamRTMPEventTypeDurationChanged**
The current streaming duration has changed.
- **bmdSwitcherStreamRTMPEventTypeAuthenticationChanged**
The service authentication credentials have changed.
- **bmdSwitcherStreamRTMPEventTypeLowLatencyChanged**
The low latency setting has changed.
- **bmdSwitcherStreamRTMPEventTypeSRTExtensionsChanged**
The SRT extensions have changed.
- **bmdSwitcherStreamRTMPEventTypeAvailabilityChanged**
The availability of streaming has changed.
- **bmdSwitcherStreamRTMPEventTypeProfileChanged**
The current streaming quality profile has changed.

11.2.4 Video Codec Type

BMDSwitcherVideoCodec enumerates the possible streaming video codecs.

- **BMDSwitcherVideoCodecH264**
H.264 (AVC) codec.
- **BMDSwitcherVideoCodecH265**
H.265 (HEVC) code.

11.2.5 Streaming Down Convert Mode

BMDSwitcherStreamDownConvertMode enumerates the possible down converting modes for streaming.

- **bmdSwitcherStreamDownConvertModeNone**
No forced down conversion.
- **bmdSwitcherStreamDownConvertModeHD1080**
Down convert to HD, 1080 pixels high.

11.3 Interface Reference

11.3.1 IBMDSwitcherStreamRTMP Interface

The **IBMDSwitcherStreamRTMP** object interface provides functionality to start and stop streaming of video and audio to a streaming server.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An IBMDSwitcherStreamRTMP object interface can be obtained with IBMDSwitcher::QueryInterface .

Public Member Functions	
Method	Description
StartStreaming	Start streaming.
StopStreaming	Stop streaming.
IsStreaming	Determine whether the switcher is currently streaming.
GetStatus	Get the current streaming status.
CanStreamRTMP	Determine whether the switcher is capable of streaming with RTMP.
CanStreamRTMPS	Determine whether the switcher is capable of streaming with RTMPS.
CanStreamSRT	Determine whether the switcher is capable of streaming with SRT.
CanStream4K	Determine whether the switcher is capable of streaming in 4K video standard.
CanDownConvert	Determine whether the switcher is capable of down converted streaming.
SetServiceName	Set the streaming service name.
GetServiceName	Get the streaming service name.
SetUrl	Set the streaming URL.
GetUrl	Get the streaming URL.
SetKey	Set the streaming key.
GetKey	Get the streaming key.
SetVideoBitrates	Set the maximum video streaming bitrates.
GetVideoBitrates	Get the maximum video streaming bitrates.
SetAudioBitrates	Set the maximum audio streaming bitrates.
GetAudioBitrates	Get the maximum audio streaming bitrates.
SetDownConvertMode	Set the streaming down convert mode.
GetDownConvertMode	Get the streaming down convert mode.
RequestDuration	Request the current streaming duration and timecode from the switcher.
GetDuration	Get the cached streaming duration (in frames).
GetTimecode	Get the cached streaming timecode.

Public Member Functions	
Method	Description
GetEncodingBitrate	Get the current encoding bitrate.
GetCacheUsed	Get the current usage level of the streaming cache.
SetAuthentication	Set the streaming authentication credentials.
GetAuthentication	Get the streaming authentication credentials.
SetLowLatency	Set the low latency flag.
GetLowLatency	Get the low latency flag.
SetSRTExtensions	Set the SRT stream extensions.
GetSRTExtensions	Get the SRT stream extensions.
SetProfileXml	Set the streaming profile.
GetProfile	Get the current streaming profile.
AddCallback	Add a streaming callback.
RemoveCallback	Remove a streaming callback.

11.3.1.1 **IBMDSwitcherStreamRTMP::StartStreaming** method

The **StartStreaming** method starts video and audio streaming to the configured streaming server.

Syntax

```
HRESULT StartStreaming()
```

Return Values

Value	Description
S_OK	Success.
S_FALSE	Already streaming.
E_FAIL	Failure.

11.3.1.2 **IBMDSwitcherStreamRTMP::StopStreaming** method

The **StopStreaming** method stops video and audio streaming to the configured streaming server.

Syntax

```
HRESULT StopStreaming()
```

Return Values

Value	Description
S_OK	Success.
S_FALSE	Not currently streaming.
E_FAIL	Failure.

11.3.1.3 IBMDSwitcherStreamRTMP::IsStreaming method

The **IsStreaming** method is used to determine if the switcher is currently streaming.

Syntax

```
HRESULT IsStreaming(Boolean* streaming)
```

Parameters

Name	Direction	Description
streaming	out	Boolean value indicating whether the switcher is currently streaming.

Return Values

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

11.3.1.4 IBMDSwitcherStreamRTMP::GetStatus method

The **GetStatus** method returns the current streaming status.

Syntax

```
HRESULT GetStatus(BMDSwitcherStreamRTMPState* state,  
BMDSwitcherStreamRTMPError* error)
```

Parameters

Name	Direction	Description
state	out	BMDSwitcherRecordAVState value indicating the current streaming status.
error	out	BMDSwitcherStreamRTMPError value indicating the error associated with current streaming status.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

11.3.1.5 IBMDSwitcherStreamRTMP::CanStreamRTMP method

The **CanStreamRTMP** method is used to determine whether the switcher is able to stream with RTMP.

Syntax

```
HRESULT CanStreamRTMP(Boolean* canStreamRTMP)
```

Parameters

Name	Direction	Description
canStreamRTMP	out	Boolean value indicating whether the switcher is able to stream with RTMP.

Return Values

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

11.3.1.6 IBMDSwitcherStreamRTMP::CanStreamRTMPS method

The **CanStreamRTMPS** method is used to determine whether the switcher is able to stream with RTMPS.

Syntax

```
HRESULT CanStreamRTMPS(Boolean* canStreamRTMPS)
```

Parameters

Name	Direction	Description
canStreamRTMPS	out	Boolean value indicating whether the switcher is able to stream with RTMPS.

Return Values

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

11.3.1.7 IBMDSwitcherStreamRTMP::CanStreamSRT method

The **CanStreamSRT** method is used to determine whether the switcher is able to stream with SRT.

Syntax

```
HRESULT CanStreamSRT(Boolean* canStreamSRT)
```

Parameters

Name	Direction	Description
canStreamSRT	out	Boolean value indicating whether the switcher is able to stream with SRT.

Return Values

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

11.3.1.8 **IBMDSwitcherStreamRTMP::CanStream4K** method

The **CanStream4K** method is used to determine whether the switcher is able to stream in 4K video standard.

Syntax

```
HRESULT CanStream4K(Boolean* canStream4K)
```

Parameters

Name	Direction	Description
canStream4K	out	Boolean value indicating whether the switcher is able to stream in 4K video standard.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The canStream4K parameter is invalid.

11.3.1.9 **IBMDSwitcherStreamRTMP::CanDownConvert** method

The **CanDownConvert** method is used to determine whether the switcher is capable of streaming at a lower resolution video standard than the video standard it is operating in.

Syntax

```
HRESULT CanDownConvert(Boolean* canDownConvert)
```

Parameters

Name	Direction	Description
canDownConvert	out	Boolean value indicating whether the switcher is able to down convert its streaming output.

Return Values

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

11.3.1.10 **IBMDSwitcherStreamRTMP::SetServiceName** method

The **SetServiceName** method sets the streaming service name. The name is only used for display purposes.

Syntax

```
HRESULT SetServiceName(string serviceName)
```

Parameters

Name	Direction	Description
serviceName	in	Name of the streaming service.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.11 **IBMDSwitcherStreamRTMP::GetServiceName** method

The **GetServiceName** method gets the name of the streaming service.

Syntax

```
HRESULT GetServiceName(string* serviceName)
```

Parameters

Name	Direction	Description
serviceName	out	Name of the streaming service.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The serviceName parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to get the service name.

11.3.1.12 **IBMDSwitcherStreamRTMP::SetUrl** method

The **SetUrl** method sets the streaming server URL.

Syntax

```
HRESULT SetUrl(string url)
```

Parameters

Name	Direction	Description
url	in	Streaming server URL.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.13 **IBMDSwitcherStreamRTMP::GetUrl** method

The **GetUrl** method gets the streaming server URL.

Syntax

```
HRESULT GetUrl(string* url)
```

Parameters

Name	Direction	Description
url	out	Streaming server URL.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The url parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to get the URL.

11.3.1.14 **IBMDSwitcherStreamRTMP::SetKey** method

The **SetKey** method sets the streaming server key.

Syntax

```
HRESULT SetKey(string url)
```

Parameters

Name	Direction	Description
url	in	Streaming server key.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.15 **IBMDSwitcherStreamRTMP::GetKey** method

The **GetKey** method gets the streaming server key.

Syntax

```
HRESULT GetKey(string* key)
```

Parameters

Name	Direction	Description
key	out	Streaming server key.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The key parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to get the key.

11.3.1.16 **IBMDSwitcherStreamRTMP::SetVideoBitrates** method

The **SetVideoBitrate** method sets the maximum video streaming bitrates, in bits per second. The low bitrate is used for framerates of 30 and lower. The high bitrate is used for framerates of 50 and higher.

Syntax

```
HRESULT SetVideoBitrates(uint32_t lowBitrate, uint32_t highBitrate)
```

Parameters

Name	Direction	Description
lowBitrate	in	Maximum video streaming bitrate for low framerates, in bits per second.
highBitrate	in	Maximum video streaming bitrate for high framerates, in bits per second.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.17 **IBMDSwitcherStreamRTMP::GetVideoBitrates** method

The **GetVideoBitrate** method gets the current maximum video streaming bitrates, in bits per second. The low bitrate is used for framerates of 30 and lower. The high bitrate is used for framerates of 50 and higher.

Syntax

```
HRESULT GetVideoBitrates(uint32_t* lowBitRate, uint32_t* highBitRate)
```

Parameters

Name	Direction	Description
lowBitRate	out	Maximum video streaming bitrate for low framerates, in bits per second.
highBitRate	out	Maximum video streaming bitrate for high framerates, in bits per second.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

11.3.1.18 **IBMDSwitcherStreamRTMP::SetAudioBitrates** method

The **SetAudioBitrate** method sets the maximum audio streaming bitrates, in bits per second. The low bitrate is used for framerates of 30 and lower. The high bitrate is used for framerates of 50 and higher.

Syntax

```
HRESULT SetAudioBitrates(uint32_t lowBitrate, uint32_t highBitrate)
```

Parameters

Name	Direction	Description
lowBitrate	in	Maximum audio streaming bitrate for low framerates, in bits per second.
highBitrate	in	Maximum audio streaming bitrate for high framerates, in bits per second.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.19 **IBMDSwitcherStreamRTMP::GetAudioBitrates** method

The **GetAudioBitrate** method gets the current maximum audio streaming bitrates, in bits per second. The low bitrate is used for framerates of 30 and lower. The high bitrate is used for framerates of 50 and higher.

Syntax

```
HRESULT GetAudioBitrates(uint32_t* lowBitRate, uint32_t* highBitRate)
```

Parameters

Name	Direction	Description
lowBitRate	out	Maximum audio streaming bitrate for low framerates, in bits per second.
highBitRate	out	Maximum audio streaming bitrate for high framerates, in bits per second.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

11.3.1.20 **IBMDSwitcherStreamRTMP::RequestDuration** method

The **RequestDuration** method requests the current streaming duration and timecode from the switcher which will be cached when received. Use the **GetDuration** and **GetTimecode** methods to get the cached duration and cached timecode, respectively.

Syntax

```
HRESULT RequestDuration()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.21 **IBMDSwitcherStreamRTMP::GetDuration** method

The **GetDuration** method returns the streaming duration (in frames) that was last received from the switcher.

Syntax

```
HRESULT GetDuration(uint64_t* duration);
```

Parameters

Name	Direction	Description
duration	out	Recording duration (in frames).

Return Values

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

11.3.1.22 **IBMDSwitcherStreamRTMP::GetTimecode** method

The **GetTimecode** method returns the streaming timecode that was last received from the switcher.

Syntax

```
HRESULT GetTimecode(uint8_t* hours, uint8_t* minutes, uint8_t* seconds, uint8_t* frames, Boolean* dropFrame);
```

Parameters

Name	Direction	Description
hours	out	The hours value of the timecode.
minutes	out	The minutes value of the timecode.
seconds	out	The seconds value of the timecode.
frames	out	The frames value of the timecode.
dropFrame	out	Whether the timecode is drop frame.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

11.3.1.23 **IBMDSwitcherStreamRTMP::GetEncodingBitrate** method

The **GetEncodingBitrate** method returns the current encoding bitrate, in bits per second.

Syntax

```
HRESULT GetEncodingBitrate(uint32_t* encodingBitrate)
```

Parameters

Name	Direction	Description
encodingBitrate	out	The current encoding bitrate.

Return Values

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

11.3.1.24 **IBMDSwitcherStreamRTMP::GetCacheUsed** method

The **GetCacheUsed** method returns the current usage level of the streaming cache, as a value with range 0.0 to 1.0.

Syntax

```
HRESULT GetCacheUsed(double* cacheUsed)
```

Parameters

Name	Direction	Description
cacheUsed	out	Current usage level of the streaming cache.

Return Values

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

11.3.1.25 **IBMDSwitcherStreamRTMP::SetDownConvertMode** method

The **SetStreamDownConvertMode** method sets the current down convert mode used for streaming.

Syntax

```
HRESULT SetDownConvertMode(BMDSwitcherStreamDownConvertMode downConvertMode)
```

Parameters

Name	Direction	Description
downConvertMode	in	Current down convert mode used for streaming.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.26 **IBMDSwitcherStreamRTMP::GetDownConvertMode** method

The **GetStreamDownConvertMode** method gets the current down convert mode used for streaming.

Syntax

```
HRESULT GetDownConvertMode(BMDSwitcherStreamDownConvertMode* downConvertMode)
```

Parameters

Name	Direction	Description
downConvertMode	out	Current down convert mode used for streaming.

Return Values

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

11.3.1.27 **IBMDSwitcherStreamRTMP::SetAuthentication** method

The **SetAuthentication** method sets the streaming server authentication credentials.

Syntax

```
HRESULT SetAuthentication(string username, string password)
```

Parameters

Name	Direction	Description
username	in	Streaming server authentication username.
password	in	Streaming server authentication password.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.28 **IBMDSwitcherStreamRTMP::GetAuthentication** method

The **GetAuthentication** method gets the streaming server authentication credentials.

Syntax

```
HRESULT GetAuthentication(string* username, string* password)
```

Parameters

Name	Direction	Description
username	out	Streaming server authentication username.
password	out	Streaming server authentication password.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The username and/or password parameters are invalid.
E_OUTOFMEMORY	Out of memory, could not assign the username and/or password parameters.

11.3.1.29 **IBMDSwitcherStreamRTMP::SetLowLatency** method

The **SetLowLatency** method changes the low latency mode. If low latency is enabled, frames will be dropped during times of poor network performance to prevent the stream from lagging too far behind the live output of the switcher. This flag can only be changed when the switcher is not recording.

Syntax

```
HRESULT SetLowLatency(Boolean lowLatency)
```

Parameters

Name	Direction	Description
lowLatency	in	Boolean value indicating whether low latency is active.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.30 **IBMDSwitcherStreamRTMP::GetLowLatency** method

The **GetLowLatency** method returns the low latency mode.

Syntax

```
HRESULT GetLowLatency(Boolean* lowLatency)
```

Parameters

Name	Direction	Description
lowLatency	out	Boolean value indicating whether low latency is active.

Return Values

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

11.3.1.31 **IBMDSwitcherStreamRTMP::SetSRTExtensions** method

The **SetSRTExtensions** method sets the SRT stream extensions.

Syntax

```
HRESULT SetSRTExtensions(string extensions)
```

Parameters

Name	Direction	Description
extensions	in	SRT stream extensions.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.32 **IBMDSwitcherStreamRTMP::GetSRTExtensions** method

The **GetSRTExtensions** method gets the SRT stream extensions.

Syntax

```
HRESULT GetSRTExtensions(string* extensions)
```

Parameters

Name	Direction	Description
extensions	out	SRT stream extensions.

Return Values

Value	Description
S_OK	Success.
E_POINTER	The extensions parameter is invalid.
E_OUTOFMEMORY	Insufficient memory to get the SRT stream ID extensions.

11.3.1.33 IBMDSwitcherStreamRTMP::SetProfileXml method

The **SetProfileXml** method sets the streaming profile using the XML text format. The streaming profile specifies encoder settings for different video standards. If the passed in XML is malformed it will be ignored. An example profile in XML format follows.

```
<profile>
  <name>Streaming High</name>
  <low-latency/>
  <config resolution="720p" fps="60" codec="H264">
    <bitrate>6000000</bitrate>
    <audio-bitrate>128000</audio-bitrate>
    <keyframe-interval>2</keyframe-interval>
  </config>
  <config resolution="720p" fps="30" codec="H264">
    <bitrate>4000000</bitrate>
    <audio-bitrate>128000</audio-bitrate>
    <keyframe-interval>2</keyframe-interval>
  </config>
  <config resolution="1080p" fps="60" codec="H264">
    <bitrate>9000000</bitrate>
    <audio-bitrate>128000</audio-bitrate>
    <keyframe-interval>2</keyframe-interval>
  </config>
  <config resolution="1080p" fps="30" codec="H264">
    <bitrate>6000000</bitrate>
    <audio-bitrate>128000</audio-bitrate>
    <keyframe-interval>2</keyframe-interval>
  </config>
</profile>
```

Syntax

```
HRESULT SetProfileXml(string xml)
```

Parameters

Name	Direction	Description
xml	in	Streaming profile in XML format.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.1.34 **BMDSwitcherStreamRTMP::GetProfile** method

The **GetStreamingProfile** method returns a full description of the current streaming profile.

Syntax

```
HRESULT GetProfile(string* profileName,  
uint32_t* lowVideoBitrates,  
uint32_t* highVideoBitrates,  
uint32_t* lowAudioBitrates,  
uint32_t* highAudioBitrates,  
Boolean* lowLatency,  
BMDSwitcherVideoMode* videoMode,  
BMDSwitcherVideoCodec* videoCodec,  
uint32_t* keyFrameInterval)
```

Parameters

Name	Direction	Description
profileName	out	Name of the profile.
lowVideoBitrates	out	Maximum video streaming bitrate for low framerates, in bits per second.
highVideoBitrates	out	Maximum video streaming bitrate for high framerates, in bits per second.
lowAudioBitrates	out	Maximum audio streaming bitrate for low framerates, in bits per second.
highAudioBitrates	out	Maximum audio streaming bitrate for high framerates, in bits per second.
lowLatency	out	Boolean value indicating whether low latency is active.
videoMode	out	The video standard used for streaming.
videoCodec	out	The video codec used for streaming.
keyFrameInterval	out	The key frame interval.

Return Values

Value	Description
S_OK	Success.
E_POINTER	One of the input parameters is invalid.
E_NOTIMPL	The switcher does not report the streaming profile.
E_OUTOFMEMORY	Insufficient memory to get the streaming profile.

11.3.1.35 **IBMDSwitcherStreamRTMP::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherStreamRTMPCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherStreamRTMPCallback object interface.

Return Values

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

11.3.1.36 **IBMDSwitcherStreamRTMP::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherStreamRTMPCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherStreamRTMPCallback object interface.

Return Values

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

11.3.2 IBMDSwitcherStreamRTMPCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherStreamRTMP	IID_IBMDSwitcherStreamRTMP	An IBMDSwitcherStreamRTMPCallback object interface is installed with IBMDSwitcherStreamRTMP::AddCallback and removed with IBMDSwitcherStreamRTMP::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.
NotifyStatus	Called when the streaming status changes.

11.3.2.1 IBMDSwitcherStreamRTMPCallback::Notify method

The **Notify** method is called when **IBMDSwitcherStreamRTMP** 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(BMDSwitcherStreamRTMPEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherStreamRTMPEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

11.3.2.2 IBMDSwitcherStreamRTMPCallback::NotifyStatus method Interface

The **NotifyStatus** method is called when the streaming status has changed.

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 NotifyStatus(BMDSwitcherStreamRTMPState stateType,  
                    BMDSwitcherStreamRTMPErr error)
```

Parameters

Name	Direction	Description
stateType	in	BMDSwitcherStreamRTMPState that describes the current streaming state.
error	in	BMDSwitcherStreamRTMPErr of the error associated with the current streaming state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 12 — Recording

The Recording API provides functionality to record video and audio from the switcher.

12.1 General Information

12.1.1 Video and Audio Encoding

Recordings from the switcher are stored as H.264 video and AAC audio in MP4 file format onto any externally connected disk (USB disk, flash drive, or Blackmagic Design MultiDock).

12.1.2 Working Set of Disks

The switcher maintains a working set of two disks. By default, this will be the two connected disks (USB disks or flash drives) with the largest available capacity. One of these will be designated as the **active** disk. The active disk is the one that will be, or is currently being, used for recording. The other disk (non active disk) will be a spill disk, which will be used for recording when the active disk is filled, or if `IBMDSwitcherRecordAV::SwitchDisk` is called. The working set can be changed by calling `IBMDSwitcherRecordAV::SetWorkingSetDisk`.

12.2 Recording Data Types

12.2.1 Recording State

`BMDSwitcherRecordAVState` enumerates the possible recording states.

- `bmdSwitcherRecordAVStateIdle`
Not recording.
- `bmdSwitcherRecordAVStateRecording`
Recording is in progress.
- `bmdSwitcherRecordAVStateStopping`
Recording is stopping.

12.2.2 Recording Error

`BMDSwitcherRecordAVError` enumerates the possible errors that can occur when recording.

- `bmdSwitcherRecordAVErrorNone`
No error
- `bmdSwitcherRecordAVErrorNoMedia`
Invalid disk or no disk.
- `bmdSwitcherRecordAVErrorMediaFull`
No space available on any disk.
- `bmdSwitcherRecordAVErrorMediaError`
Disk error.
- `bmdSwitcherRecordAVErrorMediaUnformatted`
Disk is not formatted.
- `bmdSwitcherRecordAVErrorDroppingFrames`
Frames dropped while recording.
- `bmdSwitcherRecordAVErrorUnknown`
Unknown error.

12.2.3 Recording Disk Status

BMDSwitcherRecordDiskStatus enumerates the possible recording disk statuses.

- **bmdSwitcherRecordDiskIdle**
Disk is currently not in use.
- **bmdSwitcherRecordDiskUnformatted**
Disk is not formatted.
- **bmdSwitcherRecordDiskActive**
Disk is part of the working set.
- **bmdSwitcherRecordDiskRecording**
Disk is currently being recorded to.

12.2.4 Record Disk Properties

BMDSwitcherRecordDiskProperties enumerates the possible recording disk properties.

- **bmdSwitcherRecordDiskPropertySlowDisk**
Detected disk speed is slow and may drop frames.
- **bmdSwitcherRecordDiskPropertySlowConnection**
Detected disk connection (cable) is slow and may drop frames.

12.2.5 Recording Event Type

BMDSwitcherRecordAVEventType enumerates the possible event types for the **IBMDSwitcherRecordAVCallback** object interface.

- **bmdSwitcherRecordAVEventTypeFilenameChanged**
The recording filename has changed.
- **bmdSwitcherRecordAVEventTypeRecordInAllCamerasChanged**
The Record in all Cameras state has changed.
- **bmdSwitcherRecordAVEventTypeTimecodeChanged**
Current recording timecode has changed.
- **bmdSwitcherRecordAVEventTypeDurationChanged**
Current recording duration has changed.
- **bmdSwitcherRecordAVEventTypeActiveDiskIndexChanged**
The currently active disk index has changed.
- **bmdSwitcherRecordAVEventTypeTotalRecordingTimeAvailableChanged**
The combined recording time available on all connected disks has changed.
- **bmdSwitcherRecordAVEventTypeRecordAllISOInputsChanged**
The Record all ISO Inputs state has changed.

12.2.6 Recording Disk Availability Event Type

BMDSwitcherRecordDiskAvailabilityEventType enumerates the possible disk availability event types for the **IBMDSwitcherRecordAVCallback** object interface.

- **bmdSwitcherRecordDiskAvailabilityEventTypeAvailable**
A disk has become available.
- **bmdSwitcherRecordDiskAvailabilityEventTypeRemoved**
A disk is no longer available.

12.2.7 Recording Disk Event Type

BMDSwitcherRecordDiskEventType enumerates the possible event types for the **IBMDSwitcherRecordDiskCallback** object interface.

- **bmdSwitcherRecordDiskEventTypeStatusChanged**
The disk status has changed.
- **bmdSwitcherRecordDiskEventTypeRecordingTimeAvailableChanged**
The remaining recording time on the disk has changed.
- **bmdSwitcherRecordDiskEventTypeVolumeNameChanged**
The disk volume name has changed.

12.3 Interface Reference

12.3.1 IBMDSwitcherRecordAV Interface

The **IBMDSwitcherRecordAV** object interface provides functionality to start and stop recording of video and audio to externally connected disks.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An IBMDSwitcherRecordAV object interface can be obtained with IBMDSwitcher::QueryInterface .

Public Member Functions	
Method	Description
StartRecording	Start recording.
StopRecording	Stop recording.
SwitchDisk	Switch recording to the next disk.
IsRecording	Determine whether the switcher is currently recording.
GetStatus	Get the current recording status.
SetFilename	Set the recording file name.
GetFilename	Get the recording file name.
SetRecordInAllCameras	Set the record in all cameras flag.
GetRecordInAllCameras	Get the record in all cameras flag.
DoesSupportISORecording	Determine if recording ISO inputs is supported by the switcher.

SetRecordAllISOInputs	Set the record all ISO inputs flag.
GetRecordAllISOInputs	Get the record all ISO inputs flag.
GetWorkingSetLimit	Get the maximum number of disks that can be in the working set.
SetWorkingSetDisk	Set a disk in the the working set.
GetWorkingSetDisk	Get the ID of a disk in the working set.
GetActiveDiskIndex	Get the index of the active disk.
RequestDuration	Request the current recording duration and timecode from the switcher.
GetDuration	Get the cached recording duration (in frames).
GetTimecode	Get the cached recording timecode.
GetTotalRecordingTimeAvailable	Get the total available recording time.
CreateIterator	Create an iterator.
AddCallback	Add a recording callback.
RemoveCallback	Remove a recording callback.

12.3.1.1 **IBMDSwitcherRecordAV::StartRecording method**

The **StartRecording** method starts recording video and audio to disk.

Syntax

HRESULT StartRecording()

Return Values

Value	Description
S_OK	Success.
S_FALSE	Already recording.
E_FAIL	Failure.

12.3.1.2 **IBMDSwitcherRecordAV::StopRecording method**

The **StopRecording** method stops the current recording.

Syntax

HRESULT StopRecording()

Return Values

Value	Description
S_OK	Success.
S_FALSE	Not currently recording.
E_FAIL	Failure.

12.3.1.3 **IBMDSwitcherRecordAV::SwitchDisk** method

The **SwitchDisk** method stops recording to the current disk and continues recording to the next disk, if there is one available.

Syntax

```
HRESULT SwitchDisk()
```

Return Values

Value	Description
S_OK	Success.
S_FALSE	Not currently recording.
E_FAIL	Failure.

12.3.1.4 **IBMDSwitcherRecordAV::IsRecording** method

The **IsRecording** method determines if the switcher is currently recording.

Syntax

```
HRESULT IsRecording(Boolean* recording)
```

Parameters

Name	Direction	Description
recording	out	Boolean value indicating whether the switcher is currently recording.

Return Values

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

12.3.1.5 **IBMDSwitcherRecordAV::GetStatus** method

The **GetStatus** method returns the current recording status

Syntax

```
HRESULT GetStatus(BMDSwitcherRecordAVState* state,  
BMDSwitcherRecordAVError* error)
```

Parameters

Name	Direction	Description
state	out	BMDSwitcherRecordAVState value indicating the current recording status.
error	out	BMDSwitcherRecordAVError value indicating the error associated with current recording status.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

12.3.1.6 **IBMDSwitcherRecordAV::SetFilename method**

The **SetFilename** method sets the base file name for recording. Any file extension will be ignored.

Syntax

```
HRESULT SetFilename(string filename)
```

Parameters

Name	Direction	Description
filename	in	Base name of the recording file.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.1.7 **IBMDSwitcherRecordAV::GetFilename method**

The **GetFilename** method gets the base file name for recording.

Syntax

```
HRESULT GetFilename(string* filename)
```

Parameters

Name	Direction	Description
filename	out	Base name of recording file.

Return Values

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

12.3.1.8 **IBMDSwitcherRecordAV::SetRecordInAllCameras** method

The **SetRecordInAllCameras** method changes the record in all cameras flag. If the switcher is currently recording, any connected Blackmagic Design cameras will immediately start or stop recording based on the **recordInAllCameras** flag. If the switcher is not recording, the **recordInAllCameras** flag will be used next time recording is started.

Syntax

```
HRESULT SetRecordInAllCameras(Boolean recordInAllCameras)
```

Parameters

Name	Direction	Description
recordInAllCameras	in	Boolean value indicating whether record in all cameras flag is active.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.1.9 **IBMDSwitcherRecordAV::GetRecordInAllCameras** method

The **GetRecordInAllCameras** method returns the record in all cameras flag.

Syntax

```
HRESULT GetRecordInAllCameras(Boolean* recordInAllCameras)
```

Parameters

Name	Direction	Description
recordInAllCameras	out	Boolean value indicating whether record in all cameras flag is active.

Return Values

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

12.3.1.10 **IBMDSwitcherRecordAV::DoesSupportISORecording** method

The **DoesSupportISORecording** method determines if ISO recording is supported by the switcher.

Syntax

```
HRESULT DoesSupportISORecording(Boolean* supportsISORecording)
```

Parameters

Name	Direction	Description
supportsISORecording	out	Boolean value describing whether ISO recording is supported by the switcher.

Return Values

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

12.3.1.11 **IBMDSwitcherRecordAV::SetRecordAllISOInputs** method

The **SetRecordAllISOInputs** method changes the record all ISO inputs flag. This flag can only be changed when the switcher is not recording and will be used next time recording is started.

Syntax

```
HRESULT SetRecordAllISOInputs(Boolean recordAllISOInputs)
```

Parameters

Name	Direction	Description
recordAllISOInputs	in	Boolean value indicating whether record all ISO inputs flag is active.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Feature not supported on this model, or media currently recording.

12.3.1.12 **IBMDSwitcherRecordAV::GetRecordAllISOInputs** method

The `GetRecordAllISOInputs` method returns the record all ISO inputs flag.

Syntax

```
HRESULT GetRecordAllISOInputs(BOOLEAN* recordAllISOInputs)
```

Parameters

Name	Direction	Description
recordAllISOInputs	out	Boolean value indicating whether record all ISO inputs flag is active.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Feature not supported on this model.
E_POINTER	The recordAllISOInputs parameter is invalid.

12.3.1.13 **IBMDSwitcherRecordAV::GetWorkingSetLimit** method

The `GetWorkingSetLimit` method returns the maximum number of disks that can be added to the working set.

Syntax

```
HRESULT GetWorkingSetLimit(UINT32* workingSetLimit)
```

Parameters

Name	Direction	Description
workingSetLimit	out	Number of disks in the working set.

Return Values

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

12.3.1.14 **BMDSwitcherRecordAV::SetWorkingSetDisk** method

The **SetWorkingDisk** method adds the specified disk to the working set at the specified index.

Syntax

```
HRESULT SetWorkingSetDisk(uint32_t workingSetIndex,  
                          BMDSwitcherRecordDiskId diskId)
```

Parameters

Name	Direction	Description
workingSetIndex	in	Zero based index into the working set.
diskId	in	BMDSwitcherRecordDiskId of the disk.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The workingSetIndex parameter is out of range.
E_FAIL	Failure.

12.3.1.15 **BMDSwitcherRecordAV::GetWorkingSetDisk** method

The **GetWorkingDisk** method gets the ID of the disk at the specified index in the working set.

Syntax

```
HRESULT GetWorkingSetDisk(uint32_t workingSetIndex,  
                          BMDSwitcherRecordDiskId* diskId)
```

Parameters

Name	Direction	Description
workingSetIndex	in	Zero based index into the working set
diskId	out	BMDSwitcherRecordDiskId of the disk.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The workingSetIndex parameter is out of range.
E_POINTER	The diskId parameter is invalid.
E_FAIL	Failure.

12.3.1.16 **IBMDSwitcherRecordAV::GetActiveDiskIndex** method

The **GetActiveDisk** method returns the index of the currently active disk in the working set.

Syntax

```
HRESULT GetActiveDiskIndex(uint32_t* workingSetIndex)
```

Parameters

Name	Direction	Description
workingSetIndex	out	Zero based index into the working set.

Return Values

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

12.3.1.17 **IBMDSwitcherRecordAV::RequestDuration** method

The **RequestDuration** method requests the current recording duration and timecode from the switcher which will be cached when received. Use the **GetDuration** and **GetTimeCode** methods to get the cached duration and cached timecode, respectively.

Syntax

```
HRESULT RequestDuration()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.1.18 **IBMDSwitcherRecordAV::GetDuration** method

The **GetDuration** method returns the recording duration (in frames) that was last received from the switcher.

Syntax

```
HRESULT GetDuration(uint64_t* duration);
```

Parameters

Name	Direction	Description
duration	out	Recording duration (in frames).

Return Values

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

12.3.1.19 **IBMDSwitcherRecordAV::GetTimecode** method

The **GetTimecode** method returns the recording timecode that was last received from the switcher.

Syntax

```
HRESULT GetTimecode(uint8_t* hours, uint8_t* minutes, uint8_t* seconds, uint8_t* frames, Boolean* dropFrame);
```

Parameters

Name	Direction	Description
hours	out	The hours value of the timecode.
minutes	out	The minutes value of the timecode.
seconds	out	The seconds value of the timecode.
frames	out	The frames value of the timecode.
dropFrame	out	Whether the timecode is drop frame.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.

12.3.1.20 **IBMDSwitcherRecordAV::GetTotalRecordingTimeAvailable** method

The **GetTotalRecordingTimeAvailable** method returns the total recording time available across all externally connected disks..

Syntax

```
HRESULT GetTotalRecordingTimeAvailable(uint32_t* totalRecordingTimeAvailable)
```

Parameters

Name	Direction	Description
totalRecordingTimeAvailable	out	Total recording time available across all disks.

Return Values

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

12.3.1.21 **IBMDSwitcherRecordAV::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	out	Pointer to returned 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.
E_FAIL	Failure.

12.3.1.22 **IBMDSwitcherRecordAV::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherRecordAVCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRecordAVCallback object interface.

Return Values

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

12.3.1.23 **IBMDSwitcherRecordAV::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherRecordAVCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRecordAVCallback object interface.

Return Values

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

12.3.2 IBMDSwitcherRecordAVCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRecordAV	IID_IBMDSwitcherRecordAV	An IBMDSwitcherRecordAVCallback object interface is installed with IBMDSwitcherRecordAV::AddCallback and removed with IBMDSwitcherRecordAV::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.
NotifyWorkingSetChange	Called when the working set changes.
NotifyDiskAvailability	Called when disk availability changes.
NotifyStatus	Called when the recording status changes.

12.3.2.1 IBMDSwitcherRecordAVCallback::Notify method

The **Notify** method is called when **IBMDSwitcherRecordAV** 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(BMDSwitcherRecordAVEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherRecordAVEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.2.2 **BMDSwitcherRecordAVCallback::NotifyWorkingSetChange** method

The **NotifyWorkingSetChange** method is called when a disk in the working set has changed.

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 NotifyWorkingSetChange(uint32_t workingSetIndex,  
                                BMDSwitcherRecordDiskId diskId)
```

Parameters

Name	Direction	Description
workingSetIndex	in	Working set index that has changed.
diskId	in	BMDSwitcherRecordDiskId of the disk associated with the working set change. A value of 0xFFFFFFFF means that no valid disk is present.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.2.3 **BMDSwitcherRecordAVCallback::NotifyDiskAvailability** method

The **NotifyDiskAvailability** method is called when the availability of a disk has changed.

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 NotifyDiskAvailability(BMDSwitcherRecordDiskAvailabilityEventType eventType,  
                                BMDSwitcherRecordDiskId diskId)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherRecordDiskAvailabilityEventType that describes the type of event that has occurred.
diskId	in	BMDSwitcherRecordDiskId of the disk associated with the availability change.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.2.4 **IBMDSwitcherRecordAVCallback::NotifyStatus** method

The **NotifyStatus** method is called when the recording status has changed.

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 NotifyStatus(BMDSwitcherRecordAVState stateType,  
                    BMDSwitcherRecordAVERror error)
```

Parameters

Name	Direction	Description
stateType	in	BMDSwitcherRecordAVState that describes the current recording state.
error	in	BMDSwitcherRecordAVERror of the error associated with the current recording state.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

12.3.3 **IBMDSwitcherRecordDiskIterator** Interface

The **IBMDSwitcherRecordDiskIterator** is used to enumerate the disks connected to the Switcher.

A reference to an **IBMDSwitcherRecordDiskIterator** object interface may be obtained from an **IBMDSwitcherRecordAV** object interface using the **CreateIterator** method.

Pass **IID_IBMDSwitcherRecordDiskIterator** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRecordAV	IID_IBMDSwitcherRecordAV	IBMDSwitcherRecordAV::CreateIterator can return an IBMDSwitcherRecordDiskIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherRecordDisk object interface.
GetById	Returns a pointer to an IBMDSwitcherRecorder object interface, given its BMDSwitcherRecordDiskId .

12.3.3.1 IBMDSwitcherRecordDiskIterator::Next method

The `Next` method returns the next available `IBMDSwitcherRecordDisk` object interface.

Syntax

```
HRESULT Next(IBMDSwitcherRecordDisk** recordDisk)
```

Parameters

Name	Direction	Description
recordDisk	out	IBMDSwitcherRecordDisk object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more <code>IBMDSwitcherRecordDisk</code> objects available.
E_POINTER	The recordDisk parameter is invalid.

12.3.3.2 IBMDSwitcherRecordDiskIterator::GetById method

The `GetById` method returns a pointer to an `IBMDSwitcherRecordDisk` object interface, given its `BMDSwitcherRecordDiskId`.

Syntax

```
HRESULT GetById(BMDSwitcherRecordDiskId diskId, IBMDSwitcherRecordDisk** disk)
```

Parameters

Name	Direction	Description
diskId	in	<code>BMDSwitcherRecordDiskId</code> identifier.
disk	out	IBMDSwitcherRecordDisk object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The diskId is not a valid identifier.
E_POINTER	The disk parameter is invalid.

12.3.4 IBMDSwitcherRecordDisk Interface

The `IBMDSwitcherRecordDisk` object interface provides access to information about externally connected disks.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherRecordDiskIterator</code>	<code>IID_IBMDSwitcherRecordDiskIterator</code>	An <code>IBMDSwitcherRecordDisk</code> object interface will be returned after a successful call to <code>IBMDSwitcherRecordDiskIterator::Next</code> and <code>IBMDSwitcherRecordDiskIterator::GetById</code> methods.

Public Member Functions	
Method	Description
<code>GetId</code>	Get the ID of this <code>IBMDSwitcherRecordDisk</code> interface.
<code>GetVolumeName</code>	Get the volume name.
<code>GetRecordingTimeAvailable</code>	Get the available recording time.
<code>GetStatus</code>	Get the current status.
<code>AddCallback</code>	Add a record disk callback.
<code>RemoveCallback</code>	Remove a record disk callback.

12.3.4.1 IBMDSwitcherRecordDisk::GetId method

The `GetId` method returns the disk's ID, used to uniquely identify a disk within the switcher.

Syntax

```
HRESULT GetId(BMDSwitcherRecordDiskId* diskId)
```

Parameters

Name	Direction	Description
<code>diskId</code>	out	<code>BMDSwitcherRecordDiskId</code> identifier for the current disk.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>diskId</code> parameter is invalid.

12.3.4.2 **IBMDSwitcherRecordDisk::GetVolumeName** method

The **GetVolumeName** method returns the volume name of the disk.

Syntax

```
HRESULT GetVolumeName(string* volumeName)
```

Parameters

Name	Direction	Description
volumeName	out	The volume name of the disk.

Return Values

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

12.3.4.3 **IBMDSwitcherRecordDisk::GetRecordingTimeAvailable** method

The **GetRecordingTimeAvailable** method returns the available recording time for the disk. The available recording time is calculated from the configured bitrate properties.

Syntax

```
HRESULT GetRecordingTimeAvailable(uint32_t* recordingTimeAvailable)
```

Parameters

Name	Direction	Description
recordingTimeAvailable	out	The available recording time of the disk.

Return Values

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

12.3.4.4 IBMDSwitcherRecordDisk::GetStatus method

The **GetStatus** method returns the current status of the disk.

Syntax

```
HRESULT GetStatus(IBMDSwitcherRecordDiskStatus* diskStatus)
```

Parameters

Name	Direction	Description
diskStatus	out	The current status of the disk.

Return Values

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

12.3.4.5 IBMDSwitcherRecordDisk::AddCallback method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherRecordDiskCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRecordDiskCallback object interface.

Return Values

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

12.3.4.6 IBMDSwitcherRecordDisk::RemoveCallback method

The `RemoveCallback` method removes a previously installed callback.

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherRecordDiskCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the <code>IBMDSwitcherRecordDiskCallback</code> object interface.

Return Values

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

12.3.5 IBMDSwitcherRecordDiskCallback Interface

The `IBMDSwitcherRecordDiskCallback` object interface is a callback class containing methods that are called when an event occurs on an `IBMDSwitcherRecordDisk` object.

Like all callback methods, these callback methods may be called from another thread.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherRecordDisk</code>	<code>IID_IBMDSwitcherRecordDisk</code>	An <code>IBMDSwitcherRecordDiskCallback</code> object interface is installed with <code>IBMDSwitcherRecordDisk::AddCallback</code> and removed with <code>IBMDSwitcherRecordDisk::RemoveCallback</code> .

Public Member Functions	
Method	Description
<code>Notify</code>	Called when an event occurs.

12.3.5.1 IBMDSwitcherRecordDiskCallback::Notify method

The **Notify** method is called when **IBMDSwitcherRecordDisk** 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(BMDSwitcherRecordDiskEventType eventType,  
BMDSwitcherRecordDiskId diskId)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherRecordDiskEventType that describes the type of event that has occurred.
diskId	in	BMDSwitcherRecordDiskId of the disk that has changed.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 13 — Display Clock

The DisplayClock API provides functionality to display a clock overlay on an auxiliary output.

13.1 General Information

13.1.1 Availability

Where a switcher supports this feature, it is only available on auxiliary output port 1.

13.2 DisplayClock Data Types

13.2.1 DisplayClock Event Type

BMDSwitcherDisplayClockEventType enumerates the possible event types for the **IBMDSwitcherDisplayClockCallback** object interface.

- **bmdSwitcherDisplayClockEventTypeEnabledChanged**
The enabled state has changed.
- **bmdSwitcherDisplayClockEventTypeSizeChanged**
The size has changed.
- **bmdSwitcherDisplayClockEventTypeOpacityChanged**
The opacity has changed.
- **bmdSwitcherDisplayClockEventTypePositionXChanged**
The x position has changed.
- **bmdSwitcherDisplayClockEventTypePositionYChanged**
The y position has changed.
- **bmdSwitcherDisplayClockEventTypeAutoHideChanged**
The Hide When Elapsed option has changed.
- **bmdSwitcherDisplayClockEventTypeStartFromChanged**
The start time has changed.
- **bmdSwitcherDisplayClockEventTypeClockModeChanged**
The mode has changed.
- **bmdSwitcherDisplayClockEventTypeClockStateChanged**
The state has changed.
- **bmdSwitcherDisplayClockEventTypeClockTimeChanged**
The current time has changed.

13.2.2 DisplayClock State Type

BMDSwitcherDisplayClockState enumerates the possible states of the clock.

- **bmdSwitcherDisplayClockStateStopped**
The clock has stopped.
- **bmdSwitcherDisplayClockStateRunning**
The clock is running.
- **bmdSwitcherDisplayClockStateReset**
The clock has reset.

13.2.3 DisplayClock Mode Type

BMDSwitcherDisplayClockMode enumerates the possible modes for running the clock.

- **bmdSwitcherDisplayClockModeCountdown**
The clock counts down.
- **bmdSwitcherDisplayClockModeCountup**
The clock counts up.

13.3 Interface Reference

13.3.1 IBMDSwitcherDisplayClock Interface

The **IBMDSwitcherDisplayClock** object interface provides a way to configure and display a clock on an auxilliary output.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherInputAux	IID_IBMDSwitcherInputAux	An IBMDSwitcherDisplayClock object interface can be obtained with IBMDSwitcherInputAux::QueryInterface .

Public Member Functions	
Method	Description
SetEnabled	Set the enabled flag.
GetEnabled	Get the current enabled flag.
SetOpacity	Set the opacity value.
GetOpacity	Get the current opacity value.
SetSize	Set the size value.
GetSize	Get the current size value.
SetPositionX	Set the X position value.
GetPositionX	Get the current X position value.
SetPositionY	Set the Y position value.
GetPositionY	Get the current Y position value.
SetAutoHide	Set the auto hide flag.
GetAutoHide	Get the current auto hide flag.
SetStartFrom	Set the start time (in timecode).
SetStartFromFrames	Set the start time (in frames).
GetStartFrom	Get the current start time (in timecode).
SetClockMode	Set the clock mode.
GetClockMode	Get the current clock mode.
GetClockState	Get the current state.
GetClockTime	Get the cached clock time.

Public Member Functions	
Method	Description
Start	Start the clock.
Stop	Stop the clock.
Reset	Reset the clock.
RequestTime	Request the current clock time from the switcher.
AddCallback	Add a DisplayClock callback.
RemoveCallback	Remove a DisplayClock callback.

13.3.1.1 IBMDSwitcherDisplayClock::SetEnabled method

The **SetEnabled** method sets the enabled flag which determines whether the clock is shown on the output.

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.

13.3.1.2 IBMDSwitcherDisplayClock::GetEnabled method

The **GetEnabled** method returns the current enabled flag which determines whether the clock is shown on the output.

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.
E_FAIL	Failure.

13.3.1.3 IBMDSwitcherDisplayClock::SetOpacity method

The **SetOpacity** method sets the opacity of the clock as a percentage value from 0 to 100. A value of 0 is fully transparent, and a value of 100 is fully opaque.

Syntax

```
HRESULT SetOpacity(uint16_t opacity)
```

Parameters

Name	Direction	Description
opacity	in	The desired opacity value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.4 IBMDSwitcherDisplayClock::GetOpacity method

The **GetOpacity** method returns the opacity of the clock as a percentage value between 0 and 100. A value of 0 is fully transparent, and a value of 100 is fully opaque.

Syntax

```
HRESULT GetOpacity(uint16_t* opacity)
```

Parameters

Name	Direction	Description
opacity	out	The current opacity value.

Return Values

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

13.3.1.5 IBMDSwitcherDisplayClock::SetSize method

The `SetSize` method sets the size of the clock as a value from 0 to 100.

Syntax

```
HRESULT SetSize(uint16_t size)
```

Parameters

Name	Direction	Description
size	in	The desired size value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.6 IBMDSwitcherDisplayClock::GetSize method

The `GetSize` method returns the current size of the clock as a value from 0 to 100.

Syntax

```
HRESULT GetSize(uint16_t* 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.
E_FAIL	Failure.

13.3.1.7 IBMDSwitcherDisplayClock::SetPositionX method

The `SetPositionX` method sets the X position of the clock as an aspect ratio value from -16.0 to +16.0.

Syntax

```
HRESULT SetPositionX(double positionX)
```

Parameters

Name	Direction	Description
positionX	in	The desired X position value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.8 IBMDSwitcherDisplayClock::GetPositionX method

The `GetPositionX` method returns the current X position of the clock as an aspect ratio value from -16.0 to +16.0.

Syntax

```
HRESULT GetPositionX(double* positionX)
```

Parameters

Name	Direction	Description
positionX	out	The current X position value.

Return Values

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

13.3.1.9 IBMDSwitcherDisplayClock::SetPositionY method

The `SetPositionY` method sets the Y position of the clock as an aspect ratio value from -9.0 to +9.0.

Syntax

```
HRESULT SetPositionY(double positionY)
```

Parameters

Name	Direction	Description
positionY	in	The desired Y position value.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.10 IBMDSwitcherDisplayClock::GetPositionY method

The `GetPositionY` method gets the current Y position of the clock as an aspect ratio value from -9.0 to +9.0.

Syntax

```
HRESULT GetPositionY(double* positionY)
```

Parameters

Name	Direction	Description
positionY	out	The current Y position value.

Return Values

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

13.3.1.11 IBMDSwitcherDisplayClock::SetAutoHide method

The **SetAutoHide** method sets the autohide flag, which determines whether the clock should be hidden when it reaches 00:00:00. This flag is only used when the clock is in Countdown mode.

Syntax

```
HRESULT SetAutoHide(Boolean autoHide)
```

Parameters

Name	Direction	Description
autoHide	in	The desired autoHide flag.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.12 IBMDSwitcherDisplayClock::GetAutoHide method

The **GetAutoHide** method returns the current autohide flag, which determines whether the clock should be hidden when it reaches 00:00:00. This flag is only used when the clock is in Countdown mode.

Syntax

```
HRESULT GetAutoHide(Boolean* autoHide)
```

Parameters

Name	Direction	Description
autoHide	out	The current autoHide flag.

Return Values

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

13.3.1.13 IBMDSwitcherDisplayClock::SetStartFrom method

The **SetStartFrom** method sets the start time of the clock using timecode.

Syntax

```
HRESULT SetStartFrom(uint8_t hours, uint8_t minutes,  
                    uint8_t seconds, uint8_t frames)
```

Parameters

Name	Direction	Description
hours	in	The hours value of the start time.
minutes	in	The minutes value of the start time.
seconds	in	The seconds value of the start time.
frames	in	The frames value of the start time.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.14 IBMDSwitcherDisplayClock::SetStartFromFrames method

The **SetStartFromFrames** method sets the start time of the clock using a frame count.

Syntax

```
HRESULT SetStartFromFrames(uint32_t frames)
```

Parameters

Name	Direction	Description
frames	in	The frames value of the start time.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.15 **IBMDSwitcherDisplayClock::GetStartFrom** method

The **GetStartFrom** method returns the current start time of the clock as a timecode.

Syntax

```
HRESULT GetStartFrom(uint8_t* hours, uint8_t* minutes,  
uint8_t* seconds, uint8_t* frames)
```

Parameters

Name	Direction	Description
hours	out	The hours value of the start time.
minutes	out	The minutes value of the start time.
seconds	out	The seconds value of the start time.
frames	out	The frames value of the start time.

Return Values

Value	Description
S_OK	Success.
E_POINTER	A parameter is invalid.
E_FAIL	Failure.

13.3.1.16 **IBMDSwitcherDisplayClock::SetClockMode** method

The **SetClockMode** method sets the clock mode of the clock. In Countup mode the clock counts up from 00:00:00. In Countdown mode the clock counts down to 00:00:00.

Syntax

```
HRESULT SetClockMode(BMDSwitcherDisplayClockMode clockMode)
```

Parameters

Name	Direction	Description
clockMode	in	The desired clock mode.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The clockMode parameter is not a valid value.
E_FAIL	Failure.

13.3.1.17 **IBMDSwitcherDisplayClock::GetClockMode** method

The **GetClockMode** method returns the current clock mode of the clock. In Countup mode the clock counts up from 00:00:00. In Countdown mode the clock counts down to 00:00:00.

Syntax

```
HRESULT GetClockMode(BMDSwitcherDisplayClockMode* clockMode)
```

Parameters

Name	Direction	Description
clockMode	out	The current clock mode.

Return Values

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

13.3.1.18 **IBMDSwitcherDisplayClock::GetClockState** method

The **GetClockState** method returns the current state of the clock.

Syntax

```
HRESULT GetClockState(BMDSwitcherDisplayClockState* state)
```

Parameters

Name	Direction	Description
state	out	The current clock state.

Return Values

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

13.3.1.19 **IBMDSwitcherDisplayClock::Start** method

The **Start** method starts the clock running. If the clock is already running **Start** does nothing.

Syntax

```
HRESULT Start()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.20 IBMDSwitcherDisplayClock::Stop method

The **Stop** method stops the clock running. If the clock is not running **Stop** does nothing.

Syntax

```
HRESULT Stop()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.21 IBMDSwitcherDisplayClock::Reset method

The **Reset** method resets the clock to the configured start time. If the clock is running it will be stopped.

Syntax

```
HRESULT Reset()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.22 IBMDSwitcherDisplayClock::RequestTime method

The **RequestTime** method requests the current clock time from the switcher which will be cached when received. Use the **GetClockTime** method to get the cached clock time.

Syntax

```
HRESULT RequestTime()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

13.3.1.23 IBMDSwitcherDisplayClock::AddCallback method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherDisplayClockCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherDisplayClockCallback object interface.

Return Values

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

13.3.1.24 IBMDSwitcherDisplayClock::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherDisplayClockCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherDisplayClockCallback object interface.

Return Values

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

13.3.2 IBMDSwitcherDisplayClockCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherDisplayClock	IID_ IBMDSwitcherDisplayClock	An IBMDSwitcherDisplayClockCallback object interface is installed with IBMDSwitcherDisplayClock::AddCallback and removed with IBMDSwitcherDisplayClock::RemoveCallback

Public Member Functions

Method	Description
Notify	Called when an event occurs.

13.3.2.1 IBMDSwitcherDisplayClockCallback::Notify method

The **Notify** method is called when **IBMDSwitcherDisplayClock** 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(**BMDSwitcherDisplayClockEventType** eventType)

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherDisplayClockEventType that describes the type of event that has occurred.

Section 14 — Identity Information

The IdentityInformation API provides functionality to retrieve information relating to the individual identity of the switcher.

14.1 IdentityInformation Data Types

14.1.1 Identity Information Event Type

BMDSwitcherIdentityInformationEventType enumerates the possible event types for the **BMDSwitcherIdentityInformationCallback** object interface.

- **bmdSwitcherIdentityInformationEventTypeFieldsChanged**
The identity information for the switcher has changed.

14.2 Interface Reference

14.2.1 IBMDSwitcherIdentityInformation Interface

The **IBMDSwitcherIdentityInformation** object interface is used for discovering identity information of the switcher.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	An IBMDSwitcherIdentityInformation object interface will be returned after a successful call to IBMDSwitcher::QueryInterface method.

Public Member Functions	
Method	Description
GetUniqueid	Get the unique ID of the switcher.
GetIpAddress	Get the IP address of the switcher.
GetMdnsName	Get the MDNS name of the switcher.
GetDeviceName	Get the device name of the switcher.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

14.2.1.1 IBMDSwitcherIdentityInformation::GetUniqueId method

The `GetUniqueId` method returns an ID that can be used to uniquely identify the switcher.

Syntax

```
HRESULT GetUniqueId(string* uniqueId)
```

Parameters

Name	Direction	Description
uniqueId	out	Unique ID of the switcher.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The <code>uniqueId</code> parameter is invalid.

14.2.1.2 IBMDSwitcherIdentityInformation::GetIpAddress method

The `GetIpAddress` method returns the current IP address of the switcher.

Syntax

```
HRESULT GetIpAddress(string* ipAddress)
```

Parameters

Name	Direction	Description
ipAddress	out	Local IP address of the switcher.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The <code>ipAddress</code> parameter is invalid.

14.2.1.3 **IBMDSwitcherIdentityInformation::GetMdnsName** method

The **GetMdnsName** method returns the current MDNS (Bonjour) name of the switcher. This property is generated based on the name configured in ATEM Setup.

Syntax

```
HRESULT GetMdnsName(string* mdnsName)
```

Parameters

Name	Direction	Description
mdnsName	out	Local hostname of the switcher.

Return Values

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

14.2.1.4 **IBMDSwitcherIdentityInformation::GetDeviceName** method

The **GetDeviceName** method returns the model name of the switcher.

Syntax

```
HRESULT GetDeviceName(string* deviceName)
```

Parameters

Name	Direction	Description
deviceName	out	Device name of the switcher.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.
E_POINTER	The deviceName is invalid.

14.2.1.5 IBMDSwitcherIdentityInformation::AddCallback method

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

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

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherIdentityInformationCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherIdentityInformationCallback object interface.

Return Values

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

14.2.1.6 IBMDSwitcherIdentityInformation::RemoveCallback method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherIdentityInformationCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherIdentityInformationCallback object interface.

Return Values

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

14.2.2 IBMDSwitcherIdentityInformationCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherIdentityInformation</code>	<code>IID_IBMDSwitcherIdentityInformation</code>	An <code>IBMDSwitcherIdentityInformationCallback</code> object interface is installed with <code>IBMDSwitcherIdentityInformation::AddCallback</code> and removed with <code>IBMDSwitcherIdentityInformation::RemoveCallback</code> .

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

14.2.2.1 IBMDSwitcherIdentityInformationCallback::Notify method

The `Notify` method is called when `IBMDSwitcherIdentityInformation` 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(IBMDSwitcherIdentityInformationEventType eventType)
```

Parameters

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

Section 15 — Audio Routing

The Audio Routing API provides functionality to route individual audio channels from any supported input to any supported output.

15.1 General Information

15.1.1 Audio Channel Pairs

Audio routing is performed on the basis of channel pairs. Every two consecutive audio channels are paired, for instance, channels 1 and 2, channels 3 and 4, etc. It is these pairs, rather than individual audio channels, that can be routed from any supported input to any supported output. So an Audio Routing Source is a channel pair from an audio input, and an Audio Routing Output is a channel pair from an Audio Routing Output.

15.2 Audio Routing Types

15.2.1 Audio Output Identifier

BMDSwitcherAudioOutputId is an unsigned 16 bit integer type used as a unique identifier for each audio output.

15.2.2 Audio Routing Source Identifier

BMDSwitcherAudioRoutingSourceId is an unsigned 32-bit integer type used as a unique identifier for each audio routing source.

15.2.3 Audio Routing Output Identifier

BMDSwitcherAudioRoutingOutputId is an unsigned 32-bit integer type used as a unique identifier for each audio routing output.

15.2.4 Audio Channel Pair

BMDSwitcherAudioChannelPair enumerates the possible audio channel pairs.

- **bmdSwitcherAudioChannelPair_1_2**
Audio channels 1 and 2.
- **bmdSwitcherAudioChannelPair_3_4**
Audio channels 3 and 4.
- **bmdSwitcherAudioChannelPair_5_6**
Audio channels 5 and 6.
- **bmdSwitcherAudioChannelPair_7_8**
Audio channels 7 and 8.
- **bmdSwitcherAudioChannelPair_9_10**
Audio channels 9 and 10.
- **bmdSwitcherAudioChannelPair_11_12**
Audio channels 11 and 12.
- **bmdSwitcherAudioChannelPair_13_14**
Audio channels 13 and 14.
- **bmdSwitcherAudioChannelPair_15_16**
Audio channels 15 and 16.

15.2.5 Audio Internal Port Type

BMDSwitcherAudioInternalPortType enumerates the possible port types of the audio inputs.

bmdSwitcherAudioInternalPortTypeNotInternal

The port is for external audio with a physical connector.

bmdSwitcherAudioInternalPortTypeNoAudio

There is no audio on this port.

bmdSwitcherAudioInternalPortTypeTalkbackMix

The port contains Talkback Mix audio.

bmdSwitcherAudioInternalPortTypeEngineeringTalkbackMix

The port contains Engineering Talkback audio.

bmdSwitcherAudioInternalPortTypeProductionTalkbackMix

The port contains Production Talkback audio.

bmdSwitcherAudioInternalPortTypeMediaPlayer

The port contains Media Player audio.

bmdSwitcherAudioInternalPortTypeProgram

The port contains Program audio.

bmdSwitcherAudioInternalPortTypeReturn

The port contains Camera Return audio.

bmdSwitcherAudioInternalPortTypeMonitor

The port contains Monitor audio.

bmdSwitcherAudioInternalPortTypeMadi

The port contains MADI audio.

bmdSwitcherAudioInternalPortTypeAuxOut

The port contains Auxiliary Output audio.

bmdSwitcherAudioInternalPortTypeAudioAuxOut

The port contains Audio Auxiliary Out audio (ie. Studio or Control audio outputs).

bmdSwitcherAudioInternalPortTypeReturnStreaming

The port Streaming Camera Return audio.

bmdSwitcherAudioInternalPortTypeMixMinus

The port contains Mix Minus audio.

15.2.6 Audio Routing Source Event Type

BMDSwitcherAudioRoutingSourceEventType enumerates the possible event types for the

IBMDSwitcherAudioRoutingSourceCallback object interface.

- **bmdSwitcherAudioRoutingSourceEventTypeNameChanged**
The name of the audio routing source has changed.
- **bmdSwitcherAudioRoutingSourceEventTypeNameDefaultChanged**
The name of the audio routing source has changed from the default or has been changed to the default.

15.2.7 Audio Routing Output Event Type

BMDSwitcherAudioRoutingOutputEventType enumerates the possible event types for the **IBMDSwitcherAudioRoutingOutputCallback** object interface.

- **bmdSwitcherAudioRoutingOutputEventTypeNameChanged**
The name of the audio routing output has changed.
- **bmdSwitcherAudioRoutingOutputEventTypeNameDefaultChanged**
The name of the audio routing output has changed from the default or has been changed to the default.
- **bmdSwitcherAudioRoutingOutputEventTypeSourceChanged**
The source routed to the audio routing output has changed.

15.3 Interface Reference

15.3.1 IBMDSwitcherAudioRoutingSourceIterator Interface

The **IBMDSwitcherAudioRoutingSourceIterator** object interface is used to enumerate the available audio routing sources.

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::CreateIterator can return an IBMDSwitcherAudioRoutingSourceIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherAudioRoutingSource object interface.
GetById	Returns a pointer to an IBMDSwitcherAudioRoutingSource object interface, given its BMDSwitcherAudioRoutingSourceId .

15.3.1.1 IBMDSwitcherAudioRoutingSourceIteator::Next method

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

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

Syntax

```
HRESULT Next(IBMDSwitcherAudioRoutingSource** input)
```

Parameters

Name	Direction	Description
input	out	IBMDSwitcherAudioRoutingSource object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherAudioRoutingSource objects available.
E_POINTER	The input parameter is invalid.

15.3.1.2 IBMDSwitcherAudioRoutingSourceIteator::GetById method

The **GetById** method returns the **IBMDSwitcherAudioRoutingSource** object interface corresponding to the specified ID.

Syntax

```
HRESULT GetById(BMDSwitcherAudioRoutingSourceId id,  
IBMDSwitcherAudioRoutingSource** source)
```

Parameters

Name	Direction	Description
id	in	BMDSwitcherAudioRoutingSourceId of audio routing source.
source	out	IBMDSwitcherAudioRoutingSource object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The id parameter is invalid.
E_POINTER	The source parameter is invalid.

15.3.2 IBMDSwitcherAudioRoutingSource Interface

The `IBMDSwitcherAudioRoutingSource` object interface is used for managing audio routing sources.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioRoutingSourceIterator</code>	<code>IID_IBMDSwitcherAudioRoutingSourceIterator</code>	An <code>IBMDSwitcherAudioRoutingSource</code> object interface will be returned after a successful call to <code>IBMDSwitcherAudioRoutingSourceIterator::Next</code> method.

Public Member Functions

Method	Description
<code>GetId</code>	Get the ID of this <code>IBMDSwitcherAudioRoutingSource</code> interface.
<code>GetAudioInputId</code>	Get the audio input ID of this source.
<code>GetChannelPair</code>	Get the channel pair of this source.
<code>SetName</code>	Set the source name.
<code>GetName</code>	Get the current source name.
<code>GetNameDefault</code>	Determine if the name is currently set to the factory default value.
<code>GetExternalPortType</code>	Get the physical external port type.
<code>GetInternalPortType</code>	Get the internal port type.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

15.3.2.1 IBMDSwitcherAudioRoutingSource::GetId method

The `GetId` method returns the ID of the audio routing source, used to uniquely identify the audio channel pair within the switcher.

Syntax

```
HRESULT GetId(IBMDSwitcherAudioRoutingSourceId* id)
```

Parameters

Name	Direction	Description
<code>id</code>	out	Unique ID for the audio routing source.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>id</code> parameter is invalid.

15.3.2.2 **IBMDSwitcherAudioRoutingSource::GetAudioInputId** method

The `GetAudioInputId` method returns the audio input ID corresponding to this audio routing source.

Syntax

```
HRESULT GetAudioInputId(BMDSwitcherAudioInputId* id)
```

Parameters

Name	Direction	Description
id	out	Corresponding audio input ID for the audio routing source.

Return Values

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

15.3.2.3 **IBMDSwitcherAudioRoutingSource::GetChannelPair** method

The `GetChannelPair` method returns the audio channel pair corresponding to this audio routing source.

Syntax

```
HRESULT GetChannelPair(BMDSwitcherAudioChannelPair* pair)
```

Parameters

Name	Direction	Description
pair	out	Corresponding audio channel pair for the audio routing source.

Return Values

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

15.3.2.4 **IBMDSwitcherAudioRoutingSource::SetName** method

The **SetName** method sets the name of the audio routing source.

Syntax

```
HRESULT SetName(string name)
```

Parameters

Name	Direction	Description
name	in	The audio routing source name to apply.

Return Values

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

15.3.2.5 **IBMDSwitcherAudioRoutingSource::GetName** method

The **GetName** returns the current name of the audio routing source.

Syntax

```
HRESULT GetName(string* name)
```

Parameters

Name	Direction	Description
name	out	The current audio routing source name.

Return Values

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

15.3.2.6 **IBMDSwitcherAudioRoutingSource::GetIsNameDefault** method

The **GetIsNameDefault** method is used to determine whether the name of this audio routing source is set to the factory default.

Syntax

```
HRESULT GetIsNameDefault(BOOLEAN* nameDefault)
```

Parameters

Name	Direction	Description
nameDefault	out	Boolean value indicating whether the name is set to the factory default.

Return Values

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

15.3.2.7 **IBMDSwitcherAudioRoutingSource::GetExternalPortType** method

The **GetExternalPortType** method returns the external port type of the audio routing source, indicating its physical connector type.

Syntax

```
HRESULT GetExternalPortType(BMDSwitcherExternalPortType* type)
```

Parameters

Name	Direction	Description
type	out	The external port type of the audio routing source.

Return Values

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

15.3.2.8 **IBMDSwitcherAudioRoutingSource::GetInternalPortType** method

The **GetInternalPortType** method returns the internal port type of the audio routing source. This can be used to determine if this input is an external port (i.e. **bmdSwitcherAudioInternalPortTypeNotInternal**), or any of the internal port types such as talkback mix (i.e. **bmdSwitcherAudioInternalPortTypeTalkbackMix**).

Syntax

```
HRESULT GetInternalPortType(BMDSwitcherAudioInternalPortType* type)
```

Parameters

Name	Direction	Description
type	out	The internal port type of the audio routing source.

Return Values

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

15.3.2.9 **IBMDSwitcherAudioRoutingSource::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherAudioRoutingSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioRoutingSourceCallback object interface.

Return Values

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

15.3.2.10 **IBMDSwitcherAudioRoutingSource::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherAudioRoutingSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioRoutingSourceCallback object interface.

Return Values

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

15.3.3 **IBMDSwitcherAudioRoutingSourceCallback** Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioRoutingSource	IID_IBMDSwitcherAudioRoutingSource	An IBMDSwitcherAudioRoutingSourceCallback object interface is installed with IBMDSwitcherAudioRoutingSource::AddCallback and removed with IBMDSwitcherAudioRoutingSource::RemoveCallback .

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

15.3.3.1 **IBMDSwitcherAudioRoutingSourceCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherAudioRoutingSource** 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(IBMDSwitcherAudioRoutingSourceEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherAudioRoutingSourceEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

15.3.4 **IBMDSwitcherAudioRoutingOutputIterator** Interface

The **IBMDSwitcherAudioRoutingOutputIterator** object interface is used to enumerate the available audio routing sources.

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_IBMDSwitcher	IBMDSwitcher::CreateIterator can return an IBMDSwitcherAudioRoutingOutputIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherAudioRoutingSource object interface.
GetById	Returns a pointer to an IBMDSwitcherAudioRoutingSource object interface, given its IBMDSwitcherAudioRoutingSourceId .

15.3.4.1 IBMDSwitcherAudioRoutingOutputIterator::Next method

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

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

Syntax

```
HRESULT Next(IBMDSwitcherAudioRoutingOutput** input)
```

Parameters

Name	Direction	Description
input	out	IBMDSwitcherAudioRoutingOutput object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherAudioRoutingOutput objects available.
E_POINTER	The input parameter is invalid.

15.3.4.2 IBMDSwitcherAudioRoutingOutputIterator::GetById method

The **GetById** method returns the **IBMDSwitcherAudioRoutingOutput** object interface corresponding to the specified ID.

Syntax

```
HRESULT GetById(BMDSwitcherAudioRoutingOutputId id,  
IBMDSwitcherAudioRoutingOutput** output)
```

Parameters

Name	Direction	Description
id	in	BMDSwitcherAudioRoutingOutputId of audio routing output.
output	out	IBMDSwitcherAudioRoutingOutput object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The id parameter is invalid.
E_POINTER	The output parameter is invalid.

15.3.5 IBMDSwitcherAudioRoutingOutput Interface

The `IBMDSwitcherAudioRoutingOutput` object interface is used for managing audio routing outputs.

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherAudioRoutingOutputIterator</code>	<code>IID_IBMDSwitcherAudioRoutingOutputIterator</code>	An <code>IBMDSwitcherAudioRoutingOutput</code> object interface will be returned after a successful call to <code>IBMDSwitcherAudioRoutingSourceIterator::Next</code> method.

Public Member Functions

Method	Description
<code>GetId</code>	Get the ID of this <code>IBMDSwitcherAudioRoutingOutput</code> interface.
<code>GetAudioOutputId</code>	Get the audio output ID of this output.
<code>GetChannelPair</code>	Get the channel pair of this output.
<code>SetSource</code>	Set the source to route to this output.
<code>GetSource</code>	Get the current source routed to this output.
<code>SetName</code>	Set the output name.
<code>GetName</code>	Get the current output name.
<code>GetIsNameDefault</code>	Determine if the name is currently set to the factory default value.
<code>GetExternalPortType</code>	Get the physical external port type.
<code>GetInternalPortType</code>	Get the internal port type.
<code>AddCallback</code>	Add a callback.
<code>RemoveCallback</code>	Remove a callback.

15.3.5.1 IBMDSwitcherAudioRoutingOutput::GetId method

The `GetId` method returns the ID of the audio routing output, used to uniquely identify the audio channel pair within the switcher.

Syntax

```
HRESULT GetId(IBMDSwitcherAudioRoutingOutputId* id)
```

Parameters

Name	Direction	Description
<code>id</code>	out	Unique ID for audio routing output.

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_POINTER</code>	The <code>id</code> parameter is invalid.

15.3.5.2 **IBMDSwitcherAudioRoutingOutput::GetAudioOutputId** method

The `GetAudioOutputId` method returns the audio output ID corresponding to this audio routing output.

Syntax

```
HRESULT GetAudioOutputId(BMDSwitcherAudioOutputId* id)
```

Parameters

Name	Direction	Description
id	out	Corresponding audio output ID for the audio routing output.

Return Values

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

15.3.5.3 **IBMDSwitcherAudioRoutingOutput::GetChannelPair** method

The `GetChannelPair` method returns the audio channel pair corresponding to this audio routing output.

Syntax

```
HRESULT GetChannelPair(BMDSwitcherAudioChannelPair* pair)
```

Parameters

Name	Direction	Description
pair	out	Corresponding audio channel pair for the audio routing output.

Return Values

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

15.3.5.4 **IBMDSwitcherAudioRoutingOutput::SetSource** method

The `SetSource` method sets the audio routing source that will be routed to the audio routing output.

Syntax

```
HRESULT SetSource(BMDSwitcherAudioRoutingSourceId sourceId)
```

Parameters

Name	Direction	Description
sourceId	in	Source to route to this audio routing output.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

15.3.5.5 **IBMDSwitcherAudioRoutingOutput::GetSource** method

The **GetSource** method returns the ID of the audio routing source that is currently being routed to the audio routing output.

Syntax

```
HRESULT GetSource(BMDSwitcherAudioRoutingSourceId* sourceId)
```

Parameters

Name	Direction	Description
sourceId	out	Source that is currently routed to this audio routing output.

Return Values

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

15.3.5.6 **IBMDSwitcherAudioRoutingOutput::SetName** method

The **SetName** method sets the name of the audio routing output.

Syntax

```
HRESULT SetName(string name)
```

Parameters

Name	Direction	Description
name	in	The audio routing output name to apply.

Return Values

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

15.3.5.7 **IBMDSwitcherAudioRoutingOutput::GetName** method

The **GetName** returns the current name of the audio routing output.

Syntax

```
HRESULT GetName(string* name)
```

Parameters

Name	Direction	Description
name	out	The current audio routing output name.

Return Values

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

15.3.5.8 **IBMDSwitcherAudioRoutingOutput::GetIsNameDefault** method

The **GetIsNameDefault** method is used to determine whether the name of this audio routing output is set to the factory default.

Syntax

```
HRESULT GetIsNameDefault(Boolean* nameDefault)
```

Parameters

Name	Direction	Description
nameDefault	out	Boolean value indicating whether the name is set to the factory default.

Return Values

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

15.3.5.9 **IBMDSwitcherAudioRoutingOutput::GetExternalPortType** method

The **GetExternalPortType** method returns the external port type of the audio routing output, indicating its physical connector type.

Syntax

```
HRESULT GetExternalPortType(BMDSwitcherExternalPortType* type)
```

Parameters

Name	Direction	Description
type	out	The external port type of the audio routing output.

Return Values

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

15.3.5.10 **IBMDSwitcherAudioRoutingOutput::GetInternalPortType** method

The **GetInternalPortType** method returns the internal port type of the audio routing output. This can be used to determine if this input is an external port (i.e. **bmdSwitcherAudioInternalPortTypeNotInternal**), or any of the internal port types such as talkback mix (i.e. **bmdSwitcherAudioInternalPortTypeTalkbackMix**).

Syntax

```
HRESULT GetInternalPortType(BMDSwitcherAudioInternalPortType* type)
```

Parameters

Name	Direction	Description
type	out	The internal port type of the audio routing output.

Return Values

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

15.3.5.11 **IBMDSwitcherAudioRoutingOutput::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherAudioRoutingOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioRoutingOutputCallback object interface.

Return Values

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

15.3.5.12 **IBMDSwitcherAudioRoutingOutput::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherAudioRoutingOutputCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherAudioRoutingOutputCallback object interface.

Return Values

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

15.3.6 IBMDSwitcherAudioRoutingOutputCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherAudioRoutingOutput	IID_ IBMDSwitcherAudioRoutingOutput	An IBMDSwitcherAudioRoutingOutputCallback object interface is installed with IBMDSwitcherAudioRoutingOutput::AddCallback and removed with IBMDSwitcherAudioRoutingOutput::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.

15.3.6.1 IBMDSwitcherAudioRoutingOutputCallback::Notify method

The **Notify** method is called when **IBMDSwitcherAudioRoutingOutputCallback** 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(BMDSwitcherAudioRoutingOutputEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherAudioRoutingOutputEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

Section 16 — Remote Sources

The Remote Sources API provides functionality to configure and activate network streaming sources for switchers that are capable of accepting remote source input. There are two distinct types of sources, local and external which are explained below.

16.1 General Information

16.1.1 Remote Source Input Assignment and Activation

When a remote source is connected to the switcher, it must be assigned to one of the existing inputs before it can be activated and put on air. Once a remote source is assigned to an input and activated, the switcher will ignore video input from the physical connection (HDMI or SDI) and instead use the video from the remote source.

Any number of remote sources can be connected to the switcher, but only those remote sources that are assigned (linked) to an input, and activated will be available for switching.

16.1.2 Local Remote Sources

As long as the switcher is configured to be discoverable, it will use mDNS to announce itself as a streaming server on the local network. These announcements contain streaming server details, including the streaming key. Any streaming device (camera, Web Presenter, etc) that is on the same local network will be able to use the information in the server announcements to configure itself to deliver its video stream to the switcher.

16.1.3 External Remote Sources

Streaming devices outside of the local network will not be able to receive the switcher's streaming server announcements, and so must be manually configured to connect to the switcher. External remote sources can be added and removed using this API. Each remote source will have its own streaming key in order to identify allowed devices and restrict access from unknown devices. The switcher will generate XML text for each of these remote sources, which can be saved and imported into a remote source device to allow it to connect and stream video to the switcher.

16.1.4 Linking Remote Sources

When a remote source is available, it must be linked to a physical input so that it can be put on air. Once linked, the audio and video from the remote source is used instead of the audio and video from the linked physical input. In essence, a remote source replaces a physical input, so that putting that switcher input on air will put the remote source on air. For instance, if a remote source from a camera is linked to switcher input three, then any time input three goes on air, the remote source is used, and whatever is physically attached to the input three connector (via HDMI or SDI) is ignored. Unlinking the remote source will restore input coming from the physical connector.

16.2 Remote Source Types

16.2.1 Remote Source Configuration Source Event Type

BMDSwitcherRemoteSourceConfigurationSourceEventType enumerates the possible event types for the **IBMDSwitcherRemoteSourceConfigurationCallback** object interface.

- **bmdSwitcherRemoteSourceConfigurationEventSourceAdded**
A remote source was added.
- **bmdSwitcherRemoteSourceConfigurationEventSourceRemoved**
A remote source was removed.

16.2.2 Remote Source Internet Status Type

BMDSwitcherRemoteSourceInternetStatusType enumerates the possible internet statuses resulting from an internet accessibility probe.

- **BMDSwitcherRemoteSourceInternetStatusAccessible**
The switcher is accessible from the internet.
- **BMDSwitcherRemoteSourceInternetStatusTesting**
An internet accessibility probe is in progress.
- **BMDSwitcherRemoteSourceInternetStatusOffline**
The switcher is not connected to the internet.
- **BMDSwitcherRemoteSourceInternetStatusInaccessible**
The switcher is not accessible from the internet.

16.2.3 Remote Source Port Status Type

BMDSwitcherRemoteSourcePortStatusType enumerates the possible port statuses resulting from an internet accessibility probe.

- **BMDSwitcherRemoteSourcePortStatusMapped**
The external internet port is configured and available.
- **BMDSwitcherRemoteSourcePortStatusNoGateway**
There is no gateway for the external internet port.
- **BMDSwitcherRemoteSourcePortStatusDisabled**
The external internet port is disabled.
- **BMDSwitcherRemoteSourcePortStatusConfiguring**
Configuration of the external internet port is in progress.
- **BMDSwitcherRemoteSourcePortStatusFailed**
Could not configure the external internet port.

16.2.4 Remote Source Configuration Event Type

BMDSwitcherRemoteSourceConfigurationEventType enumerates the possible event types for the **IBMDSwitcherRemoteSourceConfigurationCallback** object interface.

- **bmdSwitcherRemoteSourceConfigurationEventDiscoverableChanged**
The discoverable state of the switcher has changed.
- **bmdSwitcherRemoteSourceConfigurationEventInternetProbeStatusChanged**
The internet accessibility of the switcher has changed.
- **bmdSwitcherRemoteSourceConfigurationEventInternetSettingsChanged**
The internet settings of the switcher has changed.

16.2.5 Remote Source Event Type

BMDSwitcherRemoteSourceEventType enumerates the possible event types for the **IBMDSwitcherRemoteSourceCallback** object interface.

- **bmdSwitcherRemoteSourceEventTypeChanged**
The properties of the remote source have changed.
- **bmdSwitcherRemoteSourceEventTypeLinkChanged**
The linked input for the remote source has changed

16.2.6 Remote Source External Event Type

BMDSwitcherRemoteSourceExternalEventType enumerates the possible event types for the **IBMDSwitcherRemoteSourceExternalCallback** object interface.

- **bmdSwitcherRemoteSourceExternalEventTypeChanged**
The properties of the external remote source properties has changed.
- **bmdSwitcherRemoteSourceExternalEventTypeXMLUpdated**
The XML configuration for the external remote source has changed.

16.3 Interface Reference

16.3.1 IBMDSwitcherRemoteSourceConfiguration Interface

The `IBMDSwitcherRemoteSourceConfiguration` object interface provides functionality to configure external streaming sources, enable local streaming server discovery, and manipulate internet accessibility.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_ IBMDSwitcher	An <code>IBMDSwitcherRemoteSourceConfiguration</code> object interface can be obtained with <code>IBMDSwitcher::QueryInterface</code> .

Public Member Functions	
Method	Description
AddExternalSource	Add an external source.
RemoveExternalSource	Remove an external source.
GetDiscoverable	Get the current discoverable flag.
SetDiscoverable	Set the discoverable flag.
ForceInternetProbe	Start an internet probe.
GetInternetAddress	Get the current external internet address.
SetInternetAddress	Set the external internet address.
GetInternetPort	Get the current external internet port.
SetInternetPort	Set the external internet port.
GetInternetStatus	Get the current internet status.
GetPortStatus	Get the current internet port status.
GetMaxSources	Get the number of remote sources allowed.
CreateIterator	Create an iterator.
AddCallback	Add a remote source configuration callback.
RemoveCallback	Remove a remote source configuration callback.

16.3.1.1 **IBMDSwitcherRemoteSourceConfiguration::AddExternalSource** method

The **AddExternalSource** method adds a remote source to the list of available remote sources. Adding a remote source does not affect previously added remote sources.

Syntax

```
HRESULT AddExternalSource(string key, string name, Boolean enabled)
```

Parameters

Name	Direction	Description
key	in	A 64 byte unicode string used to identify the remote source. If this is an empty string the switcher will automatically generate a new key.
name	in	Remote source name, limited to a 64 byte unicode string.
enabled	in	A boolean flag allowing the remote source to stream to this switcher.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.2 **IBMDSwitcherRemoteSourceConfiguration::RemoveExternalSource** method

The **RemoveExternalSource** method removes a previously added remote source from the list of available remote sources.

Syntax

```
HRESULT RemoveExternalSource(uint32_t sourceId)
```

Parameters

Name	Direction	Description
sourceId	in	Source ID of the remote source, obtained from IBMDSwitcherRemoteSource::GetSourceId() .

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.3 **IBMDSwitcherRemoteSourceConfiguration::GetDiscoverable** method

The **GetDiscoverable** method gets the current discoverable flag. This flag determines if the switcher will broadcast its presence as a remote streaming server, over the network via mDNS. This allows local network streaming source devices to automatically connect to the switcher.

Syntax

```
HRESULT GetDiscoverable(Boolean* discoverable)
```

Parameters

Name	Direction	Description
discoverable	out	The current discoverable flag.

Return Values

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

16.3.1.4 **IBMDSwitcherRemoteSourceConfiguration::SetDiscoverable** method

The **SetDiscoverable** method is used to set the discoverable flag. This flag determines if the switcher will broadcast its presence as a remote streaming server, over the network via mDNS. This allows local network streaming source devices to automatically connect to the switcher.

Syntax

```
HRESULT SetDiscoverable(Boolean discoverable)
```

Parameters

Name	Direction	Description
discoverable	in	The discoverable state to apply.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.5 **IBMDSwitcherRemoteSourceConfiguration::ForceInternetProbe** method

The **ForceInternetProbe** method is used to start an internet probe. An internet probe will attempt to determine the current internet accessibility state. Once complete use **GetInternetAddress()**, **GetInternetPort()**, and **GetInternetStatus()** to retrieve the results of the probe.

Syntax

```
HRESULT ForceInternetProbe()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.6 **IBMDSwitcherRemoteSourceConfiguration::GetInternetAddress** method

The **GetInternetAddress** method is used to retrieve the current external internet address of the switcher as set by calling **SetInternetAddress()**, or as discovered by calling **ForceInternetProbe()**. This is used when generating streaming source configuration info with **IBMDSwitcherRemoteSourceExternal::GetStreamingXML()**.

Syntax

```
HRESULT GetInternetAddress(string* internetAddress)
```

Parameters

Name	Direction	Description
internetAddress	out	The current external internet address.

Return Values

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

16.3.1.7 **IBMDSwitcherRemoteSourceConfiguration::SetInternetAddress** method

The **SetInternetAddress** method is used to set the external internet address of the switcher. This is used when generating streaming source configuration info with **IBMDSwitcherRemoteSourceExternal::GetStreamingXML()**.

Syntax

```
HRESULT SetInternetAddress(string internetAddress)
```

Parameters

Name	Direction	Description
internetAddress	in	The external internet address to apply.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.8 **IBMDSwitcherRemoteSourceConfiguration::GetInternetPort** method

The **GetInternetPort** method is used to retrieve the current external internet port of the switcher as set by calling **SetInternetPort()**, or as discovered by calling **ForceInternetProbe()**. This is used when generating streaming source configuration info with **IBMDSwitcherRemoteSourceExternal::GetStreamingXML()**.

Syntax

```
HRESULT GetInternetPort(uint16_t* internetPort)
```

Parameters

Name	Direction	Description
internetPort	out	The current external internet port.

Return Values

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

16.3.1.9 **IBMDSwitcherRemoteSourceConfiguration::SetInternetPort** method

The **SetInternetPort** method is used to set the external internet port of the switcher. This is used when generating streaming source configuration in the info with **IBMDSwitcherRemoteSourceExternal::GetStreamingXML()**.

Syntax

```
HRESULT SetInternetPort(uint16_t internetPort)
```

Parameters

Name	Direction	Description
internetPort	in	The external internet port to apply.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.1.10 **IBMDSwitcherRemoteSourceConfiguration::GetInternetStatus** method

The **GetInternetStatus** method is used to retrieve the current external internet status of the switcher. This information can be helpful in configuring your local network so that remote sources on the internet (external) can stream video to the switcher.

Syntax

```
HRESULT GetInternetStatus(BMDSwitcherRemoteSourceInternetStatusType* internetStatus)
```

Parameters

Name	Direction	Description
internetStatus	out	The current external internet status of the switcher.

Return Values

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

16.3.1.11 **IBMDSwitcherRemoteSourceConfiguration::GetPortStatus** method

The **GetPortStatus** method is used to retrieve the current internet port status of the switcher as determined from a call to **ForceInternetProbe()**.

Syntax

```
HRESULT GetPortStatus(BMDSwitcherRemoteSourcePortStatusType* portStatus)
```

Parameters

Name	Direction	Description
portStatus	out	The current port status.

Return Values

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

16.3.1.12 **IBMDSwitcherRemoteSourceConfiguration::GetMaxSources** method

The **GetMaxSources** method is used to determine the maximum number of remote sources that can be linked to inputs at any one time.

Syntax

```
HRESULT GetMaxSources(uint8_t* maxRemoteSources)
```

Parameters

Name	Direction	Description
maxRemoteSources	out	The number of remote sources allowed.

Return Values

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

16.3.1.13 **IBMDSwitcherRemoteSourceConfiguration::CreateIterator** method

The **CreateIterator** method creates an iterator object interface for the specified interface ID, such as **IBMDSwitcherRemoteSource**.

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 returned 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.

16.3.1.14 **IBMDSwitcherRemoteSourceConfiguration::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherRemoteSourceConfigurationCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceConfigurationCallback object interface.

Return Values

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

16.3.1.15 **IBMDSwitcherRemoteSourceConfiguration::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherRemoteSourceConfigurationCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceConfigurationCallback object interface.

Return Values

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

16.3.2 **IBMDSwitcherRemoteSourceConfigurationCallback** Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRemoteSourceConfiguration	IID_IBMDSwitcherRemoteSourceConfiguration	An IBMDSwitcherRemoteSourceConfigurationCallback object interface is installed with IBMDSwitcherRemoteSourceConfiguration::AddCallback and removed with IBMDSwitcherRemoteSourceConfiguration::RemoveCallback .

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
NotifySource	Called when a remote source is added or removed.

16.3.2.1 **IBMDSwitcherRemoteSourceConfigurationCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherRemoteSourceConfiguration** events occur, such as discoverable status 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(BMDSwitcherRemoteSourceConfigurationEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherRemoteSourceConfigurationEventType that describes the type of event that has occurred.

16.3.2.2 IBMDSwitcherRemoteSourceConfigurationCallback::NotifySource method

The **NotifySource** method is called when a **IBMDSwitcherRemoteSource** is added or removed.

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 NotifySource(IBMDSwitcherRemoteSourceConfigurationSourceEventType  
sourceEventType,  
uint32_t sourceId)
```

Parameters

Name	Direction	Description
sourceEventType	in	IBMDSwitcherRemoteSourceConfigurationSourceEventType that describes the type of event that has occurred.
sourceId	in	Source ID of the remote source the event has occurred on.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.3 IBMDSwitcherRemoteSourceIterator Interface

The **IBMDSwitcherRemoteSourceIterator** object interface is used to enumerate the available remote sources for the switcher.

A reference to an **IBMDSwitcherRemoteSourceIterator** object interface may be obtained from an **IBMDSwitcherRemoteSourceConfiguration** object interface using the **CreateIterator** method. Pass **IID_IBMDSwitcherRemoteSourceConfiguration** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRemoteSourceConfiguration	IID_IBMDSwitcherRemoteSourceConfiguration	IBMDSwitcherRemoteSourceConfiguration::CreateIterator can return an IBMDSwitcherRemoteSourceIterator object interface.

Public Member Functions	
Method	Description
Next	Returns a pointer to the next IBMDSwitcherRemoteSource object interface.

16.3.3.1 IBMDSwitcherRemoteSourceIterator::Next method

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

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

Syntax

```
HRESULT Next(IBMDSwitcherRemoteSource** remoteSource)
```

Parameters

Name	Direction	Description
remoteSource	out	IBMDSwitcherRemoteSource object interface

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more remote sources are available.
E_POINTER	The remoteSource parameter is invalid.

16.3.4 IBMDSwitcherRemoteSource Interface

The **IBMDSwitcherRemoteSource** object interface provides functionality to manage remote sources.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRemoteSourceIterator	IID_IBMDSwitcherRemoteSourceIterator	An IBMDSwitcherRemoteSource object interface can be obtained with IBMDSwitcherRemoteSourceIterator::Next .

Public Member Functions	
Method	Description
GetSourceId	Get the ID of the remote source.
IsLinked	Determine if an input is currently linked.
GetLinkedInput	Get the current linked input.
SetLinkedInput	Set the linked input.
UnlinkInput	Unlink the current linked input.
GetName	Get the name of the remote source.
GetModelName	Get the device model of the remote source.
GetEnabled	Get the current enabled state.
SetEnabled	Set the enabled state.
GetConnected	Get the current connected state.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

16.3.4.1 IBMDSwitcherRemoteSource::GetSourceId method

The `GetSourceId` method returns the ID of the remote source.

Syntax

```
HRESULT GetSourceId(uint32_t* sourceId)
```

Parameters

Name	Direction	Description
sourceId	out	ID of the remote source.

Return Values

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

16.3.4.2 IBMDSwitcherRemoteSource::IsLinked method

The `IsLinked` method is used to determine if the remote source is currently linked to a switcher input.

Syntax

```
HRESULT IsLinked(Boolean* linked)
```

Parameters

Name	Direction	Description
linked	out	Boolean value describing whether the remote source is linked.

Return Values

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

16.3.4.3 **IBMDSwitcherRemoteSource::GetLinkedInput** method

The **GetLinkedInput** method returns the current switcher input that is linked to the remote source. The audio and video from the remote source is used instead of the audio and video from the corresponding physical input connector.

Syntax

```
HRESULT GetLinkedInput(BMDSwitcherInputId* input)
```

Parameters

Name	Direction	Description
input	out	The current switcher input linked to the remove source.

Return Values

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

16.3.4.4 **IBMDSwitcherRemoteSource::SetLinkedInput** method

The **SetLinkedInput** sets the switcher input that is linked to the remote source. The audio and video from the remote source is used instead of the audio and video from the corresponding physical input connector.

Syntax

```
HRESULT SetLinkedInput(BMDSwitcherInputId input)
```

Parameters

Name	Direction	Description
input	in	The switcher input to link to the remote source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.4.5 **IBMDSwitcherRemoteSource::UnlinkInput** method

The **UnlinkInput** method is used to unlink the switcher input from the remote source. Audio and video from the corresponding physical input connector will be used as normal.

Syntax

```
HRESULT UnlinkInput()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.4.6 **IBMDSwitcherRemoteSource::GetName** method

The **GetName** method returns the name of the remote source.

If the remote source is local, the name comes from the remote source itself. If the remote source is external, then the name comes from the information passed into the **IBMDSwitcherRemoteSourceConfiguration::AddExternalSource()** method when the remote source was configured.

Syntax

```
HRESULT GetName(string* name)
```

Parameters

Name	Direction	Description
name	out	The name of the remote source.

Return Values

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

16.3.4.7 **IBMDSwitcherRemoteSource::GetModelName** method

The **GetModelName** method returns the model name from the remote source. This information is only valid for local remote sources.

Syntax

```
HRESULT GetModelName(string* model)
```

Parameters

Name	Direction	Description
model	out	The device model name of the remote source.

Return Values

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

16.3.4.8 **IBMDSwitcherRemoteSource::GetEnabled** method

The **GetEnabled** method is used to determine whether the remote source is currently enabled.

Syntax

```
HRESULT GetEnabled(Boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current enabled state of the remote source.

Return Values

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

16.3.4.9 IBMDSwitcherRemoteSource::SetEnabled method

The `SetEnabled` is used to enable or disable the remote source.

Syntax

```
HRESULT SetEnabled(Boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The enabled state to apply to the remote source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.4.10 IBMDSwitcherRemoteSource::GetConnected method

The `GetConnected` method is used to determine if the remote source is currently connected.

Syntax

```
HRESULT GetConnected(Boolean* connected)
```

Parameters

Name	Direction	Description
connected	out	The current connected state of the remote source.

Return Values

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

16.3.4.11 **IBMDSwitcherRemoteSource::AddCallback** method

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

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

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherRemoteSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceCallback object interface.

Return Values

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

16.3.4.12 **IBMDSwitcherRemoteSource::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherRemoteSourceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceCallback object interface.

Return Values

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

16.3.5 IBMDSwitcherRemoteSourceCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRemoteSource	IID_IBMDSwitcherRemoteSource	An IBMDSwitcherRemoteSourceCallback object interface is installed with IBMDSwitcherRemoteSource::AddCallback and removed with IBMDSwitcherRemoteSource::RemoveCallback .

Public Member Functions

Method	Description
Notify	Called when an event occurs.

16.3.5.1 IBMDSwitcherRemoteSourceCallback::Notify method

The **Notify** method is called when **IBMDSwitcherRemoteSource** 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(IBMDSwitcherRemoteSourceEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	IBMDSwitcherRemoteSourceEventType that describes the type of event that has occurred.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6 IBMDSwitcherRemoteSourceExternal Interface

The `IBMDSwitcherRemoteSourceExternal` object interface provides functionality to manage external remote sources.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherRemoteSource	IID_ IBMDSwitcherRemoteSource	An <code>IBMDSwitcherRemoteSourceExternal</code> object interface can be obtained with <code>IBMDSwitcherRemoteSource::QueryInterface</code> .

Public Member Functions	
Method	Description
GetSourceId	Get the ID of the remote source.
GetKey	Get the current streaming key.
SetKey	Set the streaming key.
GenerateKey	Generate a new streaming key.
GetName	Get the current name of the remote source.
SetName	Set the name of the remote source.
GetEnabled	Get the current enabled state.
SetEnabled	Set the enabled state.
RequestStreamingXML	Request the current streaming XML.
GetStreamingXML	Get the current streaming XML.
AddCallback	Add a callback.
RemoveCallback	Remove a callback.

16.3.6.1 IBMDSwitcherRemoteSourceExternal::GetSourceId method

The `GetSourceId` method returns the ID of the remote source.

Syntax

```
HRESULT GetSourceId(uint32_t* sourceId)
```

Parameters

Name	Direction	Description
sourceId	out	ID of the remote source.

Return Values

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

16.3.6.2 **IBMDSwitcherRemoteSourceExternal::GetKey** method

The **GetKey** method returns the current streaming key for the remote source. Only streaming devices that connect with this streaming key will be allowed to stream to the switcher for this remote source.

Syntax

HRESULT GetKey(string* key)

Parameters

Name	Direction	Description
key	out	The streaming key as a string.

Return Values

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

16.3.6.3 **IBMDSwitcherRemoteSourceExternal::SetKey** method

The **SetKey** method sets the streaming key for the remote source. Only streaming devices that connect with this streaming key will be allowed to stream to the switcher for this remote source.

Syntax

HRESULT SetKey(string key)

Parameters

Name	Direction	Description
key	in	The streaming key as a string.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6.4 IBMDSwitcherRemoteSourceExternal::GenerateKey method

The `GenerateKey` method is used to generate a new, random streaming key for the remote source.

Syntax

```
HRESULT GenerateKey()
```

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6.5 IBMDSwitcherRemoteSourceExternal::GetName method

The `GetName` method returns the current name of the remote source.

Syntax

```
HRESULT GetName(string* name)
```

Parameters

Name	Direction	Description
name	out	The current name of the remote source.

Return Values

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

16.3.6.6 IBMDSwitcherRemoteSourceExternal::SetName method

The `SetName` method sets the name of the remote source.

Syntax

```
HRESULT SetName(string name)
```

Parameters

Name	Direction	Description
name	in	The name to apply to the remote source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6.7 IBMDSwitcherRemoteSourceExternal::GetEnabled method

The **GetEnabled** method is used to determine whether the remote source is currently enabled.

Syntax

```
HRESULT GetEnabled(Boolean* enabled)
```

Parameters

Name	Direction	Description
enabled	out	The current enabled state of the remote source.

Return Values

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

16.3.6.8 IBMDSwitcherRemoteSourceExternal::SetEnabled method

The **SetEnabled** is used to enable or disable the remote source.

Syntax

```
HRESULT SetEnabled(Boolean enabled)
```

Parameters

Name	Direction	Description
enabled	in	The enabled state to apply to the remote source.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6.9 **IBMDSwitcherRemoteSourceExternal::RequestStreamingXML** method

The **RequestStreamingXML** method will request the most up to date streaming XML for the remote source, and store the XML internally.

The streaming XML text can be large so it is only sent by the switcher upon request. Use **GetStreamingXML** to retrieve the actual streaming XML.

Syntax

HRESULT RequestStreamingXML()

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

16.3.6.10 **IBMDSwitcherRemoteSourceExternal::GetStreamingXML** method

The **GetStreamingXML** method retrieves the current streaming XML for the remote source. To ensure that the most up to date version of the streaming XML for the remote source is available, call **RequestStreamingXML**.

The XML text returned by this method contains connection information which will allow remote streaming devices to deliver video streams to the switcher, for this remote source. This XML can be saved to a file and imported into Blackmagic Switchers, Cameras, and Web Presenters that support streaming video to streaming servers.

Syntax

HRESULT GetStreamingXML(string* xml)

Parameters

Name	Direction	Description
xml	out	The streaming XML as a string.

Return Values

Value	Description
E_POINTER	The xml parameter is invalid.
S_OK	Success.
S_FALSE	The streaming XML is not yet available.
E_FAIL	Failure.

16.3.6.11 **IBMDSwitcherRemoteSourceExternal::AddCallback** method

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

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

Adding a new callback will not affect previously added callbacks.

Syntax

```
HRESULT AddCallback(IBMDSwitcherRemoteSourceExternalCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceExternalCallback object interface.

Return Values

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

16.3.6.12 **IBMDSwitcherRemoteSourceExternal::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherRemoteSourceExternalCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherRemoteSourceExternalCallback object interface.

Return Values

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

16.3.7 IBMDSwitcherRemoteSourceExternalCallback Interface

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

Related Interfaces

Interface	Interface ID	Description
<code>IBMDSwitcherRemoteSourceExternal</code>	<code>IID_IBMDSwitcherRemoteSourceExternal</code>	An <code>IBMDSwitcherRemoteSourceExternalCallback</code> object interface is installed with <code>IBMDSwitcherRemoteSourceExternal::AddCallback</code> and removed with <code>IBMDSwitcherRemoteSourceExternal::RemoveCallback</code> .

Public Member Functions

Method	Description
<code>Notify</code>	Called when an event occurs.

16.3.7.1 IBMDSwitcherRemoteSourceExternalCallback::Notify method

The `Notify` method is called when `IBMDSwitcherRemoteSourceExternal` 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(BMDSwitcherRemoteSourceExternalEventType eventType)
```

Parameters

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

Return Values

Value	Description
<code>S_OK</code>	Success.
<code>E_FAIL</code>	Failure.

Section 17 — VISCA PTZ Camera Control

The VISCA PTZ Camera Control API provides functionality to control pan/tilt/zoom cameras and camera heads that support the VISCA protocol. Such VISCA devices can be connected to ATEM Switchers via serial port or IP network.

17.1 General Information

17.1.1 RS422 Serial Port Connected Devices

VISCA devices can be connected by RS422 serial link to ATEM Switchers that have a serial port. The serial port will be a DE-9 or a RJ-11 connector, depending on the model of switcher. Up to seven VISCA devices can be connected to a single serial port by daisy chaining each device to the previous one in line. VISCA devices connected via serial port can be automatically detected by calling the **ScanSerialDevices** method in the **IBMDSwitcherVisca** interface.

17.1.2 IP Network Connected Devices

ATEM Switchers can be configured to control multiple VISCA devices connected to an IP network. VISCA devices connected to an IP network are not automatically detected and instead must be individually added by calling the **AddIPDevice** method on the **IBMDSwitcherVisca** interface.

17.1.3 VISCA Camera Input Mapping

Each VISCA device connected to an ATEM Switcher must be assigned to one of the switcher's inputs before it can be controlled from the ATEM Switchers SDK. Any pan, tilt, or zoom commands that target the assigned switcher input will be delivered to the VISCA device. A VISCA device that is not associated with a switcher input will not be controllable from the ATEM Switchers SDK.

17.2 VISCA PTZ Camera Control Data Types

17.2.1 VISCA Device ID

BMDSwitcherViscaDeviceId is an unsigned 32 bit integer type that is used as a unique ID for each connected VISCA device.

- **BMDSwitcherViscaDeviceIdNone**
BMDSwitcherViscaDeviceId ID indicating no VISCA device.

17.2.2 VISCA Event Type

BMDSwitcherViscaEventType enumerates the possible event types for the **IBMDSwitcherViscaCallback** object interface.

- **BMDSwitcherViscaEventTypeDeviceAdded**
A VISCA device was added.
- **BMDSwitcherViscaEventTypeDeviceRemoved**
A VISCA device was removed.
- **BMDSwitcherViscaEventTypeIPPingResponse**
A VISCA IP device ping request has concluded.

17.2.3 VISCA Device Connection Type

BMDSwitcherViscaDeviceConnectionType enumerates the possible VISCA device connection types.

- **BMDSwitcherViscaDeviceConnectionTypeIP**
VISCA over IP network.
- **BMDSwitcherViscaDeviceConnectionTypeSerial**
VISCA over RS422 serial.

17.2.4 VISCA Device State

BMDSwitcherViscaDeviceState enumerates the possible VISCA device states.

- **BMDSwitcherViscaDeviceStateDisconnected**
The VISCA device is not connected.
- **BMDSwitcherViscaDeviceStateConnecting**
The VISCA device is currently connecting.
- **BMDSwitcherViscaDeviceStateConnected**
The VISCA device is connected and available.

17.2.5 VISCA Device Event Type

BMDSwitcherViscaDeviceEventType enumerates the possible event types for **IBMDSwitcherViscaDevice**.

- **bmdSwitcherViscaDeviceEventTypeStateChanged**
The state of the VISCA device has changed.
- **bmdSwitcherViscaDeviceEventTypeNameChanged**
The name of the VISCA device has changed.
- **bmdSwitcherViscaDeviceEventTypeEnabledChanged**
The enabled state of the VISCA device has changed.
- **bmdSwitcherViscaDeviceEventTypeURLChanged**
The URL of the VISCA device has changed.

17.2.6 VISCA Ping Result

BMDSwitcherViscaPingResult enumerates the possible results of a VISCA device ping.

- **bmdSwitcherViscaPingResultSuccess**
A VISCA device has responded at the requested address.
- **bmdSwitcherViscaPingResultTimeout**
No VISCA device was found at the requested address.
- **bmdSwitcherViscaPingResultUnreachable**
Unable to open a network connection to the requested address.
- **bmdSwitcherViscaPingResultAddressConflict**
The requested address is already in use by a registered VISCA device.

17.3 Interface Reference

17.3.1 IBMDSwitcherRemoteSourceConfiguration Interface

The **IBMDSwitcherVisca** object interface provides functionality to detect serial VISCA devices and also add and remove VISCA IP devices.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcher	IID_ IBMDSwitcher	An IBMDSwitcherVisca object interface can be obtained with IBMDSwitcher::QueryInterface .

Public Member Functions	
Method	Description
DoesSupportIP	Determines if VISCA IP devices are supported by the switcher.
DoesSupportSerial	Determines if VISCA serial devices are supported by the switcher.
MaxIPDevices	Gets the maximum number of VISCA IP devices supported by the switcher.
MaxSerialDevices	Gets the maximum number of VISCA serial devices supported by the switcher.
AddIPDevice	Add a VISCA IP device.
RemoveDevice	Remove a VISCA IP device.
PingIPAddress	Send a VISCA over IP network inquiry.
ScanSerialDevices	Detect any connected VISCA serial devices.
CreateIterator	Create an iterator.
AddCallback	Add a VISCA callback.
RemoveCallback	Remove a VISCA callback.

17.3.1.1 IBMDSwitcherVisca::DoesSupportIP method

The **DoesSupportIP** method is used to determine if the switcher supports VISCA devices connected by IP network.

Syntax

```
HRESULT DoesSupportIP(Boolean* supported)
```

Parameters

Name	Direction	Description
supported	out	Boolean value indicating whether VISCA over IP is supported by the switcher.

Return Values

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

17.3.1.2 **IBMDSwitcherVisca::DoesSupportSerial** method

DoesSupportSerial method is used to determine if the switcher supports VISCA devices connected by RS422 serial port.

Syntax

```
HRESULT DoesSupportSerial(Boolean* supported)
```

Parameters

Name	Direction	Description
supported	out	Boolean value indicating whether VISCA over RS422 serial is supported by the switcher.

Return Values

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

17.3.1.3 **IBMDSwitcherVisca::MaxIPDevices** method

The **MaxIPDevices** method returns the maximum number of VISCA IP devices supported by the switcher.

Syntax

```
HRESULT MaxIPDevices(uint32_t* maxDevices)
```

Parameters

Name	Direction	Description
maxDevices	out	Maximum number of supported VISCA IP devices.

Return Values

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

17.3.1.4 **IBMDSwitcherVisca::MaxSerialDevices** method

The **MaxSerialDevices** method returns the maximum number of VISCA serial devices supported by the switcher.

Syntax

```
HRESULT MaxSerialDevices(uint32_t* maxDevices)
```

Parameters

Name	Direction	Description
maxDevices	out	Maximum number of supported VISCA serial devices.

Return Values

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

17.3.1.5 **IBMDSwitcherVisca::AddIPDevice** method

The **AddIPDevice** method adds a VISCA IP device to the list of available VISCA devices. Adding a VISCA device does not affect previously added VISCA devices.

Syntax

HRESULT AddIPDevice(string url, string name)

Parameters

Name	Direction	Description
url	in	The IPv4 address of a VISCA IP device.
name	in	VISCA device name, limited to a 64 byte unicode string.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support VISCA over IP network control.
E_FAIL	Failure.

17.3.1.6 **IBMDSwitcherVisca::RemoveDevice** method

The **RemoveDevice** method removes a previously added VISCA IP device from the list of available VISCA devices.

Syntax

HRESULT RemoveDevice(BMDSwitcherViscaDeviceId id)

Parameters

Name	Direction	Description
id	in	ID of the VISCA device.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The id parameter is not a removable VISCA device ID.
E_POINTER	Failure.

17.3.1.7 **IBMDSwitcherVisca::PingIPAddress** method

The **PingIPAddress** method will send a VISCA over IP network inquiry to confirm that a VISCA device is present at the specified IP address. Once the inquiry is complete, a **PingResponse** notification, containing the result of the inquiry, will be called on any registered **IBMDSwitcherViscaCallback** objects.

Syntax

```
HRESULT PingIPAddress(string url)
```

Parameters

Name	Direction	Description
url	in	The IPv4 address of a VISCA IP device.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support VISCA over IP network control.
E_FAIL	Failure.

17.3.1.8 **IBMDSwitcherVisca::ScanSerialDevices** method

The **ScanSerialDevices** method detects any VISCA serial devices connected to the serial port and adds them to the list of available VISCA devices.

Syntax

```
HRESULT ScanSerialDevices()
```

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The switcher does not support VISCA over serial control.
E_FAIL	Failure.

17.3.1.9 **IBMDSwitcherVisca::CreateIterator method**

The **CreateIterator** method creates an iterator object interface for the specified interface ID, such as **IBMDSwitcherViscaDeviceIterator**.

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 returned 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.

17.3.1.10 **IBMDSwitcherVisca::AddCallback method**

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherViscaCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherViscaCallback object interface.

Return Values

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

17.3.1.11 **IBMDSwitcherVisca::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherViscaCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherViscaCallback object interface.

Return Values

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

17.3.2 **IBMDSwitcherViscaCallback** Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherVisca	IID_IBMDSwitcherVisca	An IBMDSwitcherViscaCallback object interface is installed with IBMDSwitcherVisca::AddCallback and removed with IBMDSwitcherVisca::RemoveCallback .

Public Member Functions	
Method	Description
Notify	Called when an event occurs.
PingResponse	Called when a VISCA over IP network inquiry concludes.

17.3.2.1 **IBMDSwitcherViscaCallback::Notify** method

The **Notify** method is called when **IBMDSwitcherVisca** events occur, such as a new VISCA device is added.

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(BMDSwitcherViscaEventType eventType,  
              BMDSwitcherViscaDeviceId id)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherViscaEventType that describes the type of event that has occurred.
id	in	Specifies the ID of the VISCA device for which the event was triggered.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

17.3.2.2 **IBMDSwitcherViscaCallback::PingResponse** method

The **PingResponse** method is called when a VISCA IP ping request concludes.

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 PingResponse(string url,  
                   BMDSwitcherViscaPingResult result)
```

Parameters

Name	Direction	Description
url	in	The IPv4 address of the VISCA over IP network inquiry.
result	in	The result of the VISCA over IP network inquiry.

Return Values

Value	Description
S_OK	Success.
E_FAIL	Failure.

17.3.3 IBMDSwitcherViscaDevicelerator Interface

The **IBMDSwitcherViscaDevicelerator** object interface is used to enumerate the available VISCA devices.

A reference to an **IBMDSwitcherViscaDevicelerator** object interface may be obtained from an **IBMDSwitcherViscaDevice** object interface using the **Createlerator** method.

Pass **IID_IBMDSwitcherViscaDevice** for the IID parameter.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherVisca	IID_IBMDSwitcherVisca	IBMDSwitcherVisca::Createlerator can return an IBMDSwitcherViscaDevicelerator object interface.

Public Member Functions	
Method	Description
Next	Returns the next IBMDSwitcherViscaDevice object interface.
GetById	Returns an IBMDSwitcherViscaDevice object interface, given its BMSwitcherViscaDeviceld .

17.3.3.1 IBMDSwitcherViscaDevicelerator::Next method

The Next method returns a pointer to the next available **IBMDSwitcherViscaDevice** object interface.

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

Syntax

```
HRESULT Next(IBMDSwitcherViscaDevice** device)
```

Parameters

Name	Direction	Description
device	out	IBMDSwitcherViscaDevice object interface.

Return Values

Value	Description
S_OK	Success.
S_FALSE	No more IBMDSwitcherViscaDevice objects available.
E_POINTER	The device parameter is invalid

17.3.3.2 **IBMDSwitcherViscaDeviceIterator::GetById** method

The **GetById** method returns a pointer to the **IBMDSwitcherViscaDevice** object interface corresponding to the specified ID.

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

Syntax

```
HRESULT GetById(BMDSwitcherViscaDeviceId viscaId,  
                IBMDSwitcherViscaDevice** device)
```

Parameters

Name	Direction	Description
viscaId	in	BMDSwitcherViscaDeviceId of the VISCA device.
device	out	IBMDSwitcherViscaDevice object interface.

Return Values

Value	Description
S_OK	Success.
E_INVALIDARG	The viscaId parameter is invalid.
E_POINTER	The device parameter is invalid

17.3.4 IBMDSwitcherViscaDevice Interface

The **IBMDSwitcherViscaDevice** object interface provides functionality to manage individual VISCA devices.

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherViscalterator	IID_IBMDSwitcherViscalterator	An IBMDSwitcherViscaDevice object interface will be returned after a successful call to IBMDSwitcherViscaDeviceIteator::Next method.

Public Member Functions	
Method	Description
GetId	Get the ID of the VISCA device.
GetConnectionType	Get the connection type.
GetState	Get the current state.
SetEnabled	Set the enabled flag.
GetEnabled	Get the current enabled flag.
SetName	Set the device name.
GetName	Get the current device name.
GetSerialAddress	Get the current serial address of the VISCA serial device.
SetURL	Set the URL of the VISCA IP device.
GetURL	Get the current URL of the VISCA IP device.
AddCallback	Add a VISCA device callback.
RemoveCallback	Remove a VISCA device callback.

17.3.4.1 IBMDSwitcherViscaDevice::GetId method

The **GetId** method gets the internal ID used to manage the VISCA device.

Syntax

```
HRESULT GetId(IBMDSwitcherViscaDeviceId* id)
```

Parameters

Name	Direction	Description
id	out	The ID of the VISCA device.

Return Values

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

17.3.4.2 **BMDSwitcherViscaDevice::GetConnectionType** method

The **GetConnectionType** method gets the connection type of the VISCA device.

Syntax

```
HRESULT GetConnectionType(BMDSwitcherViscaDeviceConnectionType* connection)
```

Parameters

Name	Direction	Description
connection	out	The connection type of the VISCA device.

Return Values

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

17.3.4.3 **BMDSwitcherViscaDevice::GetState** method

The **GetState** method gets the current state of the VISCA device.

Syntax

```
HRESULT GetState(BMDSwitcherViscaDeviceState* state)
```

Parameters

Name	Direction	Description
state	out	The current state.

Return Values

Value	Description
S_OK	Success.
S_FALSE	The VISCA device is in an unknown state and assumed to be disconnected.
E_POINTER	The state parameter is invalid.

17.3.4.4 **IBMDSwitcherViscaDevice::SetEnabled** method

The **SetEnabled** method sets the enabled flag which determines whether the VISCA device can be controlled.

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.

17.3.4.5 **IBMDSwitcherViscaDevice::GetEnabled** method

The **GetEnabled** method returns the current enabled flag which determines whether the VISCA device can be controlled.

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.

17.3.4.6 **IBMDSwitcherViscaDevice::SetName** method

The **SetName** method sets the name of the VISCA device.

Syntax

```
HRESULT SetName(string name)
```

Parameters

Name	Direction	Description
name	in	The VISCA device name to apply.

Return Values

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

17.3.4.7 **IBMDSwitcherViscaDevice::GetName** method

The **GetName** method returns the current name of the VISCA device.

Syntax

```
HRESULT GetName(string* name)
```

Parameters

Name	Direction	Description
name	out	The current VISCA device name.

Return Values

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

17.3.4.8 **IBMDSwitcherViscaDevice::GetSerialAddress** method

The **GetSerialAddress** method gets the serial address of the VISCA serial device.

Syntax

```
HRESULT GetSerialAddress(uint8_t* address)
```

Parameters

Name	Direction	Description
address	out	The serial address of the VISCA device.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The device is not a VISCA serial device.
E_POINTER	The address parameter is invalid.

17.3.4.9 **IBMDSwitcherViscaDevice::SetURL** method

The **SetURL** method sets the URL of the VISCA IP device.

Syntax

```
HRESULT SetURL(string url)
```

Parameters

Name	Direction	Description
url	in	The IPv4 address of the VISCA IP device.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The device is not a VISCA IP device.
E_FAIL	Failure.

17.3.4.10 **IBMDSwitcherViscaDevice::GetURL** method

The **GetURL** method returns the URL of the VISCA IP device.

Syntax

```
HRESULT GetURL(string* url)
```

Parameters

Name	Direction	Description
url	out	The IPv4 address of the VISCA IP device.

Return Values

Value	Description
S_OK	Success.
E_NOTIMPL	The device is not a VISCA IP device.
E_POINTER	The url parameter is invalid.
E_FAIL	Failure.

17.3.4.11 **IBMDSwitcherViscaDevice::AddCallback** method

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

Syntax

```
HRESULT AddCallback(IBMDSwitcherViscaDeviceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherViscaDeviceCallback object interface.

Return Values

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

17.3.4.12 **IBMDSwitcherViscaDevice::RemoveCallback** method

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

Syntax

```
HRESULT RemoveCallback(IBMDSwitcherViscaDeviceCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	Callback object implementing the IBMDSwitcherViscaDeviceCallback object interface.

Return Values

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

17.3.5 **IBMDSwitcherViscaDeviceCallback** Interface

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

Related Interfaces

Interface	Interface ID	Description
IBMDSwitcherViscaDevice	IID_IBMDSwitcherViscaDevice	An IBMDSwitcherViscaDeviceCallback object interface is installed with IBMDSwitcherDeviceVisca::AddCallback and removed with IBMDSwitcherDeviceVisca::RemoveCallback .

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

17.3.5.1 **IBMDSwitcherViscaDeviceCallback::Notify** method

The Notify method is called when **IBMDSwitcherViscaDevice** events occur, 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(BMDSwitcherViscaDeviceEventType eventType)
```

Parameters

Name	Direction	Description
eventType	in	BMDSwitcherViscaDeviceEventType that describes the type of event that has occurred.