

Halyri - Häke

0.9

Generated by Doxygen 1.8.7

Sat May 24 2014 10:35:56



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Packages . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>7</b>
3.1	Class List . . . . .	7
<b>4</b>	<b>File Index</b>	<b>9</b>
4.1	File List . . . . .	9
<b>5</b>	<b>Namespace Documentation</b>	<b>11</b>
5.1	Package GraphClass . . . . .	11
5.2	Package Hake_WPF . . . . .	11
5.2.1	Function Documentation . . . . .	12
5.2.1.1	ConnectionsUpdated . . . . .	12
5.2.1.2	MeasurementDataReceivedDelegate . . . . .	12
5.2.1.3	TcpMediaDataReceived . . . . .	12
5.2.1.4	UdpMediaDataReceived . . . . .	12
5.3	Package Hake_WPF.AudioVideoManagers . . . . .	13
5.3.1	Function Documentation . . . . .	13
5.3.1.1	JpgImageReceived . . . . .	13
5.4	Package Hake_WPF.CallCenterReference . . . . .	13
5.4.1	Enumeration Type Documentation . . . . .	14
5.4.1.1	ConnectionPriorityDto . . . . .	14
5.4.1.2	ConnectionStateDto . . . . .	14
5.4.1.3	LocationTypeDto . . . . .	14
5.4.1.4	MeasurementInstrumentTypeDto . . . . .	14
5.4.1.5	MediaTypeDto . . . . .	15
5.4.1.6	RemoteActionDto . . . . .	15
5.5	Package Hake_WPF.Conversion . . . . .	15
5.6	Package Hake_WPF.Network . . . . .	15

5.7	Package Hake_WPF.Properties	16
5.8	Package XamlGeneratedNamespace	16
<b>6</b>	<b>Class Documentation</b>	<b>17</b>
6.1	Hake_WPF.Conversion.AddressConverter Class Reference	17
6.1.1	Detailed Description	17
6.1.2	Member Function Documentation	17
6.1.2.1	Convert	17
6.1.2.2	ConvertBack	18
6.2	Hake_WPF.App Class Reference	18
6.2.1	Detailed Description	19
6.2.2	Member Function Documentation	19
6.2.2.1	InitializeComponent	19
6.2.2.2	InitializeComponent	19
6.2.2.3	Main	19
6.2.2.4	Main	19
6.3	Hake_WPF.Assignment Class Reference	19
6.3.1	Detailed Description	21
6.3.2	Member Enumeration Documentation	21
6.3.2.1	priorities	21
6.3.2.2	States	22
6.3.3	Constructor & Destructor Documentation	22
6.3.3.1	Assignment	22
6.3.3.2	Assignment	22
6.3.4	Member Function Documentation	22
6.3.4.1	DownloadStringCallback	22
6.3.4.2	getStreetNameFromLocationAsync	23
6.3.4.3	NotifyPropertyChanged	23
6.3.4.4	SetBackgroundColor	23
6.3.4.5	SetContent	23
6.3.5	Member Data Documentation	23
6.3.5.1	deviceInfo	23
6.3.5.2	emergencyType	23
6.3.5.3	guid	23
6.3.5.4	IsHandlerProperty	23
6.3.5.5	location	23
6.3.5.6	locationAccuracyMeters	23
6.3.5.7	locationAcquisitionTime	23
6.3.5.8	noSound	24
6.3.5.9	personalInfo	24

6.3.5.10	priority	24
6.3.5.11	pushpins	24
6.3.5.12	state	24
6.3.5.13	stateBrushes	24
6.3.5.14	streetName	24
6.3.5.15	textMessages	24
6.3.5.16	time	24
6.3.5.17	WebClientForLocationQuery	24
6.3.6	Property Documentation	24
6.3.6.1	DeviceInfo	24
6.3.6.2	EmergencyType	24
6.3.6.3	Guid	24
6.3.6.4	IsHandler	24
6.3.6.5	Location	24
6.3.6.6	LocationAccuracyMeters	25
6.3.6.7	LocationAcquisitionTime	25
6.3.6.8	NoSound	25
6.3.6.9	PersonallInfo	25
6.3.6.10	Priority	25
6.3.6.11	Pushpins	25
6.3.6.12	State	25
6.3.6.13	StreetName	25
6.3.6.14	TextMessages	25
6.3.6.15	Time	25
6.3.7	Event Documentation	25
6.3.7.1	PropertyChanged	25
6.4	Hake_WPF.AsyncReceiver Class Reference	26
6.4.1	Detailed Description	26
6.4.2	Member Function Documentation	26
6.4.2.1	ActiveConnectionsUpdated	26
6.4.2.2	AudioVideoReceived	27
6.4.2.3	MeasurementDataReceived	27
6.4.3	Member Data Documentation	27
6.4.3.1	MeasurementDataReceivedEvent	27
6.4.3.2	TcpMediaDataReceivedEvent	27
6.4.4	Event Documentation	27
6.4.4.1	ConnectionsUpdatedEvent	27
6.5	Hake_WPF.AudioVideoManagers.AudioVideoTransferManager Class Reference	27
6.5.1	Detailed Description	28
6.5.2	Constructor & Destructor Documentation	28

6.5.2.1	<a href="#">AudioVideoTransferManager</a>	28
6.5.3	<a href="#">Member Function Documentation</a>	29
6.5.3.1	<a href="#">DisableIncomingAudio</a>	29
6.5.3.2	<a href="#">DisableOutgoingAudio</a>	29
6.5.3.3	<a href="#">EnableIncomingAudio</a>	29
6.5.3.4	<a href="#">EnableOutgoingAudio</a>	29
6.5.3.5	<a href="#">InitializeAudioIODevices</a>	29
6.5.3.6	<a href="#">InitializeSpeex</a>	29
6.5.3.7	<a href="#">MicrophoneDataAvailable</a>	29
6.5.3.8	<a href="#">PlayWaveFragment</a>	29
6.5.3.9	<a href="#">SpeexAudioReceived</a>	30
6.5.3.10	<a href="#">TcpMediaReceivedHandler</a>	31
6.5.3.11	<a href="#">UdpMediaReceivedHandler</a>	31
6.5.4	<a href="#">Member Data Documentation</a>	31
6.5.4.1	<a href="#">AudioCaptureFormat</a>	31
6.5.4.2	<a href="#">audioPlaybackActive</a>	31
6.5.4.3	<a href="#">bufferedWaveStream</a>	31
6.5.4.4	<a href="#">connection</a>	31
6.5.4.5	<a href="#">decoder</a>	31
6.5.4.6	<a href="#">defaultSpeexQuality</a>	31
6.5.4.7	<a href="#">encoder</a>	31
6.5.4.8	<a href="#">JpgImageReceivedEvent</a>	31
6.5.4.9	<a href="#">outgoingSpeexInformation</a>	31
6.5.4.10	<a href="#">playbackConfigurationLock</a>	31
6.5.4.11	<a href="#">player</a>	31
6.5.4.12	<a href="#">recorder</a>	32
6.5.4.13	<a href="#">speexFramesInSecond</a>	32
6.5.4.14	<a href="#">useOutgoingUdp</a>	32
6.5.5	<a href="#">Property Documentation</a>	32
6.5.5.1	<a href="#">UseOutgoingUdp</a>	32
6.6	<a href="#">Hake_WPF.Conversion.BoolConverter Class Reference</a>	32
6.6.1	<a href="#">Detailed Description</a>	32
6.6.2	<a href="#">Member Function Documentation</a>	32
6.6.2.1	<a href="#">Convert</a>	32
6.6.2.2	<a href="#">ConvertBack</a>	33
6.7	<a href="#">Hake_WPF.AudioVideoManagers.BufferWaveStream Class Reference</a>	33
6.7.1	<a href="#">Detailed Description</a>	34
6.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	34
6.7.2.1	<a href="#">BufferWaveStream</a>	34
6.7.3	<a href="#">Member Function Documentation</a>	34

6.7.3.1	Read	34
6.7.3.2	Write	35
6.7.4	Member Data Documentation	36
6.7.4.1	completePcmSegments	36
6.7.4.2	defaultPosition	36
6.7.4.3	lengthInSamples	36
6.7.4.4	overflowFromLastSegment	36
6.7.4.5	queueLock	36
6.7.4.6	waveFormat	36
6.7.5	Property Documentation	36
6.7.5.1	Length	36
6.7.5.2	Position	36
6.7.5.3	WaveFormat	36
6.8	Hake_WPF.CallCenterReference.CallCenterConnectionDto Class Reference	36
6.8.1	Member Function Documentation	37
6.8.1.1	RaisePropertyChanged	37
6.8.2	Member Data Documentation	37
6.8.2.1	ConnectionGuidId	37
6.8.2.2	extensionDataField	37
6.8.2.3	UserNameField	37
6.8.3	Property Documentation	37
6.8.3.1	ConnectionGuid	37
6.8.3.2	ExtensionData	37
6.8.3.3	UserName	37
6.8.4	Event Documentation	37
6.8.4.1	PropertyChanged	37
6.9	Hake_WPF.Conversion.CompressionHelper Class Reference	37
6.9.1	Detailed Description	38
6.9.2	Member Function Documentation	38
6.9.2.1	CompressGZip	38
6.9.2.2	DecompressGZip	38
6.10	Hake_WPF.Connection Class Reference	38
6.10.1	Detailed Description	40
6.10.2	Constructor & Destructor Documentation	40
6.10.2.1	Connection	40
6.10.3	Member Function Documentation	41
6.10.3.1	AssigmentUpdated	41
6.10.3.2	CancelAudioVideoMedia	41
6.10.3.3	CancelAudioVideoMediaThread	41
6.10.3.4	CancelMeasurementData	41

6.10.3.5	<a href="#">ChangePriority</a>	41
6.10.3.6	<a href="#">ChangePriorityThread</a>	41
6.10.3.7	<a href="#">Connect</a>	42
6.10.3.8	<a href="#">ConvertConnectionsToAssignments</a>	42
6.10.3.9	<a href="#">Disconnect</a>	42
6.10.3.10	<a href="#">GetAllConnections</a>	42
6.10.3.11	<a href="#">InitializeUdpClient</a>	42
6.10.3.12	<a href="#">IsConnected</a>	42
6.10.3.13	<a href="#">PingAsync</a>	43
6.10.3.14	<a href="#">ProcessAssignment</a>	43
6.10.3.15	<a href="#">receiver_ConnectionsUpdatedEvent</a>	43
6.10.3.16	<a href="#">Reconnect</a>	43
6.10.3.17	<a href="#">RequestAction</a>	43
6.10.3.18	<a href="#">RequestAudioVideoMedia</a>	43
6.10.3.19	<a href="#">RequestAudioVideoMediaThread</a>	44
6.10.3.20	<a href="#">RequestMeasurementData</a>	44
6.10.3.21	<a href="#">SendMessage</a>	44
6.10.3.22	<a href="#">SendTextMessage</a>	44
6.10.3.23	<a href="#">UdpMediaPacketReceivedHandler</a>	44
6.10.3.24	<a href="#">UdpSendMedia</a>	44
6.10.3.25	<a href="#">UdpSocketFailedEventHandler</a>	45
6.10.3.26	<a href="#">UpdateConnectionsList</a>	45
6.10.4	<a href="#">Member Data Documentation</a>	45
6.10.4.1	<a href="#">client</a>	45
6.10.4.2	<a href="#">connectionDtos</a>	45
6.10.4.3	<a href="#">context</a>	45
6.10.4.4	<a href="#">currentConnection</a>	45
6.10.4.5	<a href="#">guid</a>	45
6.10.4.6	<a href="#">isConnected</a>	45
6.10.4.7	<a href="#">UdpMediaDataReceivedEvent</a>	45
6.10.4.8	<a href="#">udpServerIp</a>	45
6.10.4.9	<a href="#">udpServerPort</a>	45
6.10.4.10	<a href="#">udpSocket</a>	45
6.10.4.11	<a href="#">userConnection</a>	45
6.10.4.12	<a href="#">userCredentialsDto</a>	45
6.10.5	<a href="#">Property Documentation</a>	45
6.10.5.1	<a href="#">receiver</a>	45
6.10.6	<a href="#">Event Documentation</a>	45
6.10.6.1	<a href="#">AssignmentUpdatedEvent</a>	45
6.11	<a href="#">Hake_WPF.CallCenterReference.ConnectionDto Class Reference</a>	46



6.11.1	Member Function Documentation	47
6.11.1.1	RaisePropertyChanged	47
6.11.2	Member Data Documentation	47
6.11.2.1	ArrivalTimeField	47
6.11.2.2	AttachedCallCenterConnectionsField	47
6.11.2.3	clientDisconnectedField	47
6.11.2.4	ConnectionGuidField	47
6.11.2.5	ConnectionPriorityField	47
6.11.2.6	ConnectionStateField	47
6.11.2.7	EmergencyTypeField	47
6.11.2.8	extensionDataField	47
6.11.2.9	LocationInformationField	47
6.11.2.10	MeasurementInstrumentsField	47
6.11.2.11	MobileDeviceInformationField	47
6.11.2.12	PersonalInformationField	47
6.11.2.13	RequestedNoSoundField	47
6.11.2.14	TextMessagesField	47
6.11.3	Property Documentation	47
6.11.3.1	ArrivalTime	47
6.11.3.2	AttachedCallCenterConnections	47
6.11.3.3	clientDisconnected	47
6.11.3.4	ConnectionGuid	47
6.11.3.5	ConnectionPriority	48
6.11.3.6	ConnectionState	48
6.11.3.7	EmergencyType	48
6.11.3.8	ExtensionData	48
6.11.3.9	LocationInformation	48
6.11.3.10	MeasurementInstruments	48
6.11.3.11	MobileDeviceInformation	48
6.11.3.12	PersonalInformation	48
6.11.3.13	RequestedNoSound	48
6.11.3.14	TextMessages	48
6.11.4	Event Documentation	48
6.11.4.1	PropertyChanged	48
6.12	Hake_WPF.CallCenterReference.ConnectionFault Class Reference	48
6.13	Hake_WPF.CallCenterReference.EmergencyTypeDto Class Reference	49
6.13.1	Member Function Documentation	49
6.13.1.1	RaisePropertyChanged	49
6.13.2	Member Data Documentation	49
6.13.2.1	extensionDataField	49

6.13.2.2	TypeNameField	49
6.13.3	Property Documentation	49
6.13.3.1	ExtensionData	49
6.13.3.2	TypeName	49
6.13.4	Event Documentation	49
6.13.4.1	PropertyChanged	49
6.14	Hake_WPF.CallCenterReference.Fault Class Reference	50
6.14.1	Member Function Documentation	50
6.14.1.1	RaisePropertyChanged	50
6.14.2	Member Data Documentation	50
6.14.2.1	CauseField	50
6.14.2.2	DetailField	50
6.14.2.3	extensionDataField	50
6.14.3	Property Documentation	50
6.14.3.1	Cause	50
6.14.3.2	Detail	50
6.14.3.3	ExtensionData	50
6.14.4	Event Documentation	51
6.14.4.1	PropertyChanged	51
6.15	XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference	51
6.15.1	Detailed Description	51
6.15.2	Member Function Documentation	52
6.15.2.1	AddEventHandler	52
6.15.2.2	AddEventHandler	52
6.15.2.3	CreateDelegate	52
6.15.2.4	CreateDelegate	52
6.15.2.5	CreateInstance	52
6.15.2.6	CreateInstance	52
6.15.2.7	GetPropertyValue	52
6.15.2.8	GetPropertyValue	52
6.15.2.9	SetPropertyValue	52
6.15.2.10	SetPropertyValue	53
6.16	GraphClass.Graph Class Reference	53
6.16.1	Detailed Description	53
6.16.2	Constructor & Destructor Documentation	54
6.16.2.1	Graph	54
6.16.3	Member Function Documentation	55
6.16.3.1	AddPoint	55
6.16.3.2	ScrollToEnd	55
6.16.3.3	setBytesPerSecond	55

6.16.4	Member Data Documentation	55
6.16.4.1	BytesPerSecond	55
6.16.4.2	color1	55
6.16.4.3	color2	55
6.16.4.4	GraphCanvas	55
6.16.4.5	GraphColor	55
6.16.4.6	GraphContainer	55
6.16.4.7	GraphLine	55
6.16.4.8	GraphLineX	55
6.16.4.9	GraphScaleTransform	55
6.16.4.10	GraphSecondColor	55
6.16.4.11	scale	55
6.16.4.12	ScaleRate	55
6.17	Hake_WPF.CallCenterReference.IWcfCallCenterService Interface Reference	56
6.17.1	Member Function Documentation	57
6.17.1.1	Connect	57
6.17.1.2	ConnectAsync	57
6.17.1.3	Disconnect	57
6.17.1.4	DisconnectAsync	58
6.17.1.5	GetActiveConnections	58
6.17.1.6	GetActiveConnectionsAsync	58
6.17.1.7	MarkProcessedCloseConnection	58
6.17.1.8	MarkProcessedCloseConnectionAsync	58
6.17.1.9	MoveConnectionToHold	58
6.17.1.10	MoveConnectionToHoldAsync	58
6.17.1.11	OpenConnectionForProcessing	58
6.17.1.12	OpenConnectionForProcessingAsync	58
6.17.1.13	Ping	59
6.17.1.14	PingAsync	59
6.17.1.15	Reconnect	59
6.17.1.16	ReconnectAsync	59
6.17.1.17	RequestMediaDownstreaming	59
6.17.1.18	RequestMediaDownstreamingAsync	59
6.17.1.19	RequestMediaUpstreaming	59
6.17.1.20	RequestMediaUpstreamingAsync	59
6.17.1.21	RequestRemoteAction	59
6.17.1.22	RequestRemoteActionAsync	60
6.17.1.23	RequestStartMeasurement	60
6.17.1.24	RequestStartMeasurementAsync	60
6.17.1.25	RequestStopMeasurement	60

6.17.1.26 RequestStopMeasurementAsync . . . . .	60
6.17.1.27 SendTextMessage . . . . .	60
6.17.1.28 SendTextMessageAsync . . . . .	60
6.17.1.29 SetConnectionPriority . . . . .	60
6.17.1.30 SetConnectionPriorityAsync . . . . .	61
6.17.1.31 TransferConnection . . . . .	61
6.17.1.32 TransferConnectionAsync . . . . .	61
6.17.1.33 UploadMediaSegment . . . . .	61
6.17.1.34 UploadMediaSegmentAsync . . . . .	61
6.18 Hake_WPF.CallCenterReference.IWcfCallCenterServiceCallback Interface Reference . . . . .	61
6.18.1 Member Function Documentation . . . . .	62
6.18.1.1 ActiveConnectionsUpdated . . . . .	62
6.18.1.2 AudioVideoReceived . . . . .	62
6.18.1.3 MeasurementDataReceived . . . . .	62
6.19 Hake_WPF.CallCenterReference.IWcfCallCenterServiceChannel Interface Reference . . . . .	62
6.20 Hake_WPF.Conversion.LocationConverter Class Reference . . . . .	62
6.20.1 Detailed Description . . . . .	63
6.20.2 Member Function Documentation . . . . .	63
6.20.2.1 Convert . . . . .	63
6.20.2.2 ConvertBack . . . . .	63
6.21 Hake_WPF.CallCenterReference.LocationInformationDto Class Reference . . . . .	63
6.21.1 Member Function Documentation . . . . .	64
6.21.1.1 RaisePropertyChanged . . . . .	64
6.21.2 Member Data Documentation . . . . .	64
6.21.2.1 AccuracyMetersField . . . . .	64
6.21.2.2 AcquisitionTimeField . . . . .	64
6.21.2.3 extensionDataField . . . . .	64
6.21.2.4 LatitudeField . . . . .	64
6.21.2.5 LocationTypeField . . . . .	64
6.21.2.6 LongitudeField . . . . .	64
6.21.3 Property Documentation . . . . .	64
6.21.3.1 AccuracyMeters . . . . .	64
6.21.3.2 AcquisitionTime . . . . .	64
6.21.3.3 ExtensionData . . . . .	64
6.21.3.4 Latitude . . . . .	64
6.21.3.5 LocationType . . . . .	64
6.21.3.6 Longitude . . . . .	65
6.21.4 Event Documentation . . . . .	65
6.21.4.1 PropertyChanged . . . . .	65
6.22 Hake_WPF.MainWindow Class Reference . . . . .	65

6.22.1	Detailed Description	67
6.22.2	Constructor & Destructor Documentation	67
6.22.2.1	MainWindow	67
6.22.3	Member Function Documentation	67
6.22.3.1	AssignmentsListBox_SelectionChanged	67
6.22.3.2	CancelMeasurementDataFromDevice_Click	67
6.22.3.3	CenterMapButton_Click	68
6.22.3.4	ChangeAssignmentPriority_Click	68
6.22.3.5	ChatSendButton_Click	68
6.22.3.6	ChatTextBox_KeyUp	68
6.22.3.7	Connect	68
6.22.3.8	Connect	68
6.22.3.9	connection_AssignmentUpdatedEvent	68
6.22.3.10	GetMobileUserLocationButton_Click	68
6.22.3.11	GetVideoButton_Click	68
6.22.3.12	HandleAssignment_Click	68
6.22.3.13	IncomingPictureHandler	69
6.22.3.14	InitializeComponent	69
6.22.3.15	InitializeComponent	69
6.22.3.16	mapController_MapWindowClosingEvent	69
6.22.3.17	MapToOwnWindow_Click	69
6.22.3.18	MeasurementDataExpander_Collapsed	69
6.22.3.19	MeasurementDataExpander_Expanded	69
6.22.3.20	MeasurementDataReceived	69
6.22.3.21	OnClosing	70
6.22.3.22	RequestMeasurementDataFromDevice_Click	70
6.22.3.23	RotateVideoToLeftButton_Click	70
6.22.3.24	RotateVideoToRightButton_Click	70
6.22.3.25	SettingsMenuItem_Click	70
6.22.3.26	StopVideoButton_Click	70
6.22.3.27	UnHandleAssignmentButton_Click	70
6.22.4	Member Data Documentation	70
6.22.4.1	_contentLoaded	70
6.22.4.2	angle	70
6.22.4.3	assignmentActive	70
6.22.4.4	assignments	70
6.22.4.5	audioVideoManager	70
6.22.4.6	connection	70
6.22.4.7	FirstMeasurementPacket	71
6.22.4.8	isMeasurementPlaying	71

6.22.4.9	isVideoPlaying	71
6.22.4.10	mapController	71
6.22.4.11	measurementGraph	71
6.22.4.12	ownAssignment	71
6.22.4.13	settings	71
6.22.4.14	settingsWindow	71
6.23	Hake_WPF.MapController Class Reference	71
6.23.1	Detailed Description	72
6.23.2	Constructor & Destructor Documentation	72
6.23.2.1	MapController	72
6.23.3	Member Function Documentation	72
6.23.3.1	AddAllPushpins	72
6.23.3.2	AddMap	72
6.23.3.3	AddPushPinsToMap	72
6.23.3.4	AddPushPinToAssignment	73
6.23.3.5	CenterMaps	73
6.23.3.6	Close	73
6.23.3.7	MapToOwnWindow	73
6.23.3.8	mapWindow_Closing	73
6.23.3.9	MapWindowClosing	73
6.23.4	Member Data Documentation	73
6.23.4.1	maps	73
6.23.4.2	mapWindow	73
6.23.4.3	pushpins	73
6.23.5	Event Documentation	74
6.23.5.1	MapWindowClosingEvent	74
6.24	Hake_WPF.MapWindow Class Reference	74
6.24.1	Detailed Description	74
6.24.2	Constructor & Destructor Documentation	74
6.24.2.1	MapWindow	74
6.24.3	Member Function Documentation	75
6.24.3.1	Connect	75
6.24.3.2	Connect	75
6.24.3.3	InitializeComponent	75
6.24.3.4	InitializeComponent	75
6.24.4	Member Data Documentation	75
6.24.4.1	_contentLoaded	75
6.25	Hake_WPF.CallCenterReference.MeasurementInstrumentDto Class Reference	75
6.25.1	Member Function Documentation	76
6.25.1.1	RaisePropertyChanged	76

6.25.2	Member Data Documentation	76
6.25.2.1	DataHeaderLengthField	76
6.25.2.2	DataSampleChannelsField	76
6.25.2.3	DataSampleSizeField	76
6.25.2.4	DataSamplesPerSecondField	76
6.25.2.5	DeviceDescriptionField	76
6.25.2.6	DeviceIdentifierField	76
6.25.2.7	DeviceTypeField	76
6.25.2.8	extensionDataField	76
6.25.2.9	HeaderRepeatsOnEveryPacketField	76
6.25.3	Property Documentation	76
6.25.3.1	DataHeaderLength	76
6.25.3.2	DataSampleChannels	76
6.25.3.3	DataSampleSize	77
6.25.3.4	DataSamplesPerSecond	77
6.25.3.5	DeviceDescription	77
6.25.3.6	DeviceIdentifier	77
6.25.3.7	DeviceType	77
6.25.3.8	ExtensionData	77
6.25.3.9	HeaderRepeatsOnEveryPacket	77
6.25.4	Event Documentation	77
6.25.4.1	PropertyChanged	77
6.26	Hake_WPF.CallCenterReference.MediaConfigurationDto Class Reference	77
6.26.1	Member Function Documentation	78
6.26.1.1	RaisePropertyChanged	78
6.26.2	Member Data Documentation	78
6.26.2.1	AudioCompressionQualityField	78
6.26.2.2	AudioCompressionQualityMaxField	78
6.26.2.3	AudioCompressionQualityMinField	78
6.26.2.4	EnableAudioField	78
6.26.2.5	EnablePictureField	78
6.26.2.6	extensionDataField	78
6.26.2.7	PictureCompressionQualityField	78
6.26.2.8	PictureCompressionQualityMaxField	78
6.26.2.9	PictureCompressionQualityMinField	78
6.26.2.10	PictureFpsField	78
6.26.2.11	PictureResolutionField	78
6.26.3	Property Documentation	78
6.26.3.1	AudioCompressionQuality	78
6.26.3.2	AudioCompressionQualityMax	78

6.26.3.3	<a href="#">AudioCompressionQualityMin</a>	79
6.26.3.4	<a href="#">EnableAudio</a>	79
6.26.3.5	<a href="#">EnablePicture</a>	79
6.26.3.6	<a href="#">ExtensionData</a>	79
6.26.3.7	<a href="#">PictureCompressionQuality</a>	79
6.26.3.8	<a href="#">PictureCompressionQualityMax</a>	79
6.26.3.9	<a href="#">PictureCompressionQualityMin</a>	79
6.26.3.10	<a href="#">PictureFps</a>	79
6.26.3.11	<a href="#">PictureResolution</a>	79
6.26.4	<a href="#">Event Documentation</a>	79
6.26.4.1	<a href="#">PropertyChanged</a>	79
6.27	<a href="#">Hake_WPF.CallCenterReference.MedialInformationDto Class Reference</a>	79
6.27.1	<a href="#">Member Function Documentation</a>	80
6.27.1.1	<a href="#">RaisePropertyChanged</a>	80
6.27.2	<a href="#">Member Data Documentation</a>	80
6.27.2.1	<a href="#">CompressedGZipField</a>	80
6.27.2.2	<a href="#">extensionDataField</a>	80
6.27.2.3	<a href="#">MediaTypeField</a>	80
6.27.3	<a href="#">Property Documentation</a>	80
6.27.3.1	<a href="#">CompressedGZip</a>	80
6.27.3.2	<a href="#">ExtensionData</a>	80
6.27.3.3	<a href="#">MediaType</a>	80
6.27.4	<a href="#">Event Documentation</a>	80
6.27.4.1	<a href="#">PropertyChanged</a>	80
6.28	<a href="#">Hake_WPF.Conversion.MinuteConverter Class Reference</a>	80
6.28.1	<a href="#">Detailed Description</a>	81
6.28.2	<a href="#">Member Function Documentation</a>	81
6.28.2.1	<a href="#">Convert</a>	81
6.28.2.2	<a href="#">ConvertBack</a>	81
6.29	<a href="#">Hake_WPF.CallCenterReference.MobileDeviceInformationDto Class Reference</a>	81
6.29.1	<a href="#">Member Function Documentation</a>	82
6.29.1.1	<a href="#">RaisePropertyChanged</a>	82
6.29.2	<a href="#">Member Data Documentation</a>	82
6.29.2.1	<a href="#">CellularEnabledField</a>	82
6.29.2.2	<a href="#">CellularMobileOperatorField</a>	82
6.29.2.3	<a href="#">extensionDataField</a>	82
6.29.2.4	<a href="#">NetworkAvailableField</a>	82
6.29.2.5	<a href="#">RemainingChargePercentField</a>	82
6.29.2.6	<a href="#">RemainingDischargeTimeField</a>	83
6.29.2.7	<a href="#">RoamingEnabledField</a>	83



6.29.2.8	WiFiEnabledField	83
6.29.3	Property Documentation	83
6.29.3.1	CellularEnabled	83
6.29.3.2	CellularMobileOperator	83
6.29.3.3	ExtensionData	83
6.29.3.4	NetworkAvailable	83
6.29.3.5	RemainingChargePercent	83
6.29.3.6	RemainingDischargeTime	83
6.29.3.7	RoamingEnabled	83
6.29.3.8	WiFiEnabled	83
6.29.4	Event Documentation	83
6.29.4.1	PropertyChanged	83
6.30	Hake_WPF.CallCenterReference.ParameterFault Class Reference	83
6.31	Hake_WPF.CallCenterReference.PersonalInformationDto Class Reference	83
6.31.1	Member Function Documentation	84
6.31.1.1	RaisePropertyChanged	84
6.31.2	Member Data Documentation	84
6.31.2.1	extensionDataField	84
6.31.2.2	LocalityField	84
6.31.2.3	NameField	84
6.31.2.4	PhoneNumbersField	84
6.31.2.5	PostalCodeField	84
6.31.2.6	StreetAddressField	84
6.31.3	Property Documentation	84
6.31.3.1	ExtensionData	85
6.31.3.2	Locality	85
6.31.3.3	Name	85
6.31.3.4	PhoneNumbers	85
6.31.3.5	PostalCode	85
6.31.3.6	StreetAddress	85
6.31.4	Event Documentation	85
6.31.4.1	PropertyChanged	85
6.32	Hake_WPF.Conversion.PhoneNumberConverter Class Reference	85
6.32.1	Detailed Description	85
6.32.2	Member Function Documentation	85
6.32.2.1	Convert	85
6.32.2.2	ConvertBack	86
6.33	Hake_WPF.Network.Pinger Class Reference	86
6.33.1	Detailed Description	86
6.33.2	Constructor & Destructor Documentation	87

6.33.2.1	Pinger	87
6.33.3	Member Function Documentation	88
6.33.3.1	SendKeepalive	88
6.33.3.2	StartPinger	88
6.33.3.3	StopPinger	88
6.33.4	Member Data Documentation	88
6.33.4.1	connection	88
6.33.4.2	dispatcherTimer	88
6.33.4.3	keepaliveIntervalSeconds	88
6.34	Hake_WPF.Conversion.PriorityConverter Class Reference	88
6.34.1	Detailed Description	89
6.34.2	Member Function Documentation	89
6.34.2.1	Convert	89
6.34.2.2	ConvertBack	89
6.35	Hake_WPF.Settings Class Reference	89
6.35.1	Detailed Description	90
6.35.2	Constructor & Destructor Documentation	90
6.35.2.1	Settings	90
6.35.3	Member Function Documentation	90
6.35.3.1	AddOrUpdate	90
6.35.3.2	Get	90
6.35.3.3	Remove	91
6.35.3.4	Save	91
6.35.4	Member Data Documentation	91
6.35.4.1	ENDPOINTADDRESS	91
6.35.4.2	formatter	91
6.35.4.3	GRAPHBACKGROUNDCOLOR	91
6.35.4.4	GRAPHLINECOLOR	91
6.35.4.5	NEWURGENTSTATECOLOR	91
6.35.4.6	settings	91
6.35.4.7	store	91
6.36	Hake_WPF.SettingsWindow Class Reference	91
6.36.1	Detailed Description	92
6.36.2	Constructor & Destructor Documentation	92
6.36.2.1	SettingsWindow	92
6.36.3	Member Function Documentation	92
6.36.3.1	Connect	92
6.36.3.2	Connect	92
6.36.3.3	InitializeComponent	92
6.36.3.4	InitializeComponent	92

6.36.3.5	PreviewListBox_SelectionChanged . . . . .	92
6.36.3.6	SaveButton_Click . . . . .	93
6.36.3.7	StateComboBox_SelectionChanged . . . . .	93
6.36.4	Member Data Documentation . . . . .	93
6.36.4.1	_contentLoaded . . . . .	93
6.36.4.2	settings . . . . .	93
6.37	Hake_WPF.AudioVideoManagers.SpeexCompression Class Reference . . . . .	93
6.37.1	Detailed Description . . . . .	93
6.37.2	Member Function Documentation . . . . .	93
6.37.2.1	CompressSpeex . . . . .	93
6.37.2.2	DecompressSpeex . . . . .	94
6.38	Hake_WPF.Conversion.StateConverter Class Reference . . . . .	94
6.38.1	Detailed Description . . . . .	95
6.38.2	Member Function Documentation . . . . .	95
6.38.2.1	Convert . . . . .	95
6.38.2.2	ConvertBack . . . . .	95
6.38.3	Member Data Documentation . . . . .	95
6.38.3.1	states . . . . .	95
6.39	Hake_WPF.CallCenterReference.TargetStateFault Class Reference . . . . .	95
6.40	Hake_WPF.CallCenterReference.TextMessageDto Class Reference . . . . .	96
6.40.1	Member Enumeration Documentation . . . . .	96
6.40.1.1	MessageOriginatorDto . . . . .	96
6.40.2	Member Function Documentation . . . . .	97
6.40.2.1	RaisePropertyChanged . . . . .	97
6.40.3	Member Data Documentation . . . . .	97
6.40.3.1	ContentField . . . . .	97
6.40.3.2	extensionDataField . . . . .	97
6.40.3.3	OriginatorField . . . . .	97
6.40.3.4	TimeStampField . . . . .	97
6.40.4	Property Documentation . . . . .	97
6.40.4.1	Content . . . . .	97
6.40.4.2	ExtensionData . . . . .	97
6.40.4.3	Originator . . . . .	97
6.40.4.4	TimeStamp . . . . .	97
6.40.5	Event Documentation . . . . .	97
6.40.5.1	PropertyChanged . . . . .	97
6.41	Hake_WPF.CallCenterReference.UserCredentialsDto Class Reference . . . . .	97
6.41.1	Member Function Documentation . . . . .	98
6.41.1.1	RaisePropertyChanged . . . . .	98
6.41.2	Member Data Documentation . . . . .	98

6.41.2.1	<a href="#">extensionDataField</a>	98
6.41.2.2	<a href="#">PasswordField</a>	98
6.41.2.3	<a href="#">UserNameField</a>	98
6.41.3	<a href="#">Property Documentation</a>	98
6.41.3.1	<a href="#">ExtensionData</a>	98
6.41.3.2	<a href="#">Password</a>	98
6.41.3.3	<a href="#">UserName</a>	98
6.41.4	<a href="#">Event Documentation</a>	98
6.41.4.1	<a href="#">PropertyChanged</a>	98
6.42	<a href="#">Hake_WPF.CallCenterReference.WcfCallCenterServiceClient Class Reference</a>	98
6.42.1	<a href="#">Constructor &amp; Destructor Documentation</a>	100
6.42.1.1	<a href="#">WcfCallCenterServiceClient</a>	100
6.42.1.2	<a href="#">WcfCallCenterServiceClient</a>	100
6.42.1.3	<a href="#">WcfCallCenterServiceClient</a>	100
6.42.1.4	<a href="#">WcfCallCenterServiceClient</a>	100
6.42.1.5	<a href="#">WcfCallCenterServiceClient</a>	100
6.42.2	<a href="#">Member Function Documentation</a>	100
6.42.2.1	<a href="#">Connect</a>	101
6.42.2.2	<a href="#">ConnectAsync</a>	101
6.42.2.3	<a href="#">Disconnect</a>	101
6.42.2.4	<a href="#">DisconnectAsync</a>	101
6.42.2.5	<a href="#">GetActiveConnections</a>	101
6.42.2.6	<a href="#">GetActiveConnectionsAsync</a>	101
6.42.2.7	<a href="#">MarkProcessedCloseConnection</a>	101
6.42.2.8	<a href="#">MarkProcessedCloseConnectionAsync</a>	101
6.42.2.9	<a href="#">MoveConnectionToHold</a>	101
6.42.2.10	<a href="#">MoveConnectionToHoldAsync</a>	102
6.42.2.11	<a href="#">OpenConnectionForProcessing</a>	102
6.42.2.12	<a href="#">OpenConnectionForProcessingAsync</a>	102
6.42.2.13	<a href="#">Ping</a>	102
6.42.2.14	<a href="#">PingAsync</a>	102
6.42.2.15	<a href="#">Reconnect</a>	102
6.42.2.16	<a href="#">ReconnectAsync</a>	102
6.42.2.17	<a href="#">RequestMediaDownstreaming</a>	102
6.42.2.18	<a href="#">RequestMediaDownstreamingAsync</a>	102
6.42.2.19	<a href="#">RequestMediaUpstreaming</a>	103
6.42.2.20	<a href="#">RequestMediaUpstreamingAsync</a>	103
6.42.2.21	<a href="#">RequestRemoteAction</a>	103
6.42.2.22	<a href="#">RequestRemoteActionAsync</a>	103
6.42.2.23	<a href="#">RequestStartMeasurement</a>	103

6.42.2.24 RequestStartMeasurementAsync . . . . .	103
6.42.2.25 RequestStopMeasurement . . . . .	103
6.42.2.26 RequestStopMeasurementAsync . . . . .	103
6.42.2.27 SendTextMessage . . . . .	104
6.42.2.28 SendTextMessageAsync . . . . .	104
6.42.2.29 SetConnectionPriority . . . . .	104
6.42.2.30 SetConnectionPriorityAsync . . . . .	104
6.42.2.31 TransferConnection . . . . .	104
6.42.2.32 TransferConnectionAsync . . . . .	104
6.42.2.33 UploadMediaSegment . . . . .	104
6.42.2.34 UploadMediaSegmentAsync . . . . .	104
<b>7 File Documentation . . . . .</b>	<b>107</b>
7.1 AddressConverter.cs File Reference . . . . .	107
7.2 App.g.cs File Reference . . . . .	107
7.3 App.g.i.cs File Reference . . . . .	107
7.4 App.xaml.cs File Reference . . . . .	108
7.5 AssemblyInfo.cs File Reference . . . . .	108
7.6 Assignment.cs File Reference . . . . .	108
7.7 AudioVideoTransferManager.cs File Reference . . . . .	108
7.8 BoolConverter.cs File Reference . . . . .	108
7.9 BufferWaveStream.cs File Reference . . . . .	109
7.10 CompressionHelper.cs File Reference . . . . .	109
7.11 Connection.cs File Reference . . . . .	109
7.12 GeneratedInternalTypeHelper.g.cs File Reference . . . . .	110
7.13 GeneratedInternalTypeHelper.g.i.cs File Reference . . . . .	110
7.14 GraphClass.cs File Reference . . . . .	110
7.15 LocationConverter.cs File Reference . . . . .	110
7.16 MainWindow.g.cs File Reference . . . . .	111
7.17 MainWindow.g.i.cs File Reference . . . . .	111
7.18 MainWindow.xaml.cs File Reference . . . . .	111
7.19 MapController.cs File Reference . . . . .	111
7.20 MapWindow.g.cs File Reference . . . . .	112
7.21 MapWindow.g.i.cs File Reference . . . . .	112
7.22 MapWindow.xaml.cs File Reference . . . . .	112
7.23 MinuteConverter.cs File Reference . . . . .	112
7.24 PhoneNumberConverter.cs File Reference . . . . .	113
7.25 Pinger.cs File Reference . . . . .	113
7.26 PriorityConverter.cs File Reference . . . . .	113
7.27 Reference.cs File Reference . . . . .	113

7.28 Resources.Designer.cs File Reference . . . . .	114
7.29 Settings.cs File Reference . . . . .	115
7.30 Settings.Designer.cs File Reference . . . . .	115
7.31 SettingsWindow.g.cs File Reference . . . . .	115
7.32 SettingsWindow.g.i.cs File Reference . . . . .	115
7.33 SettingsWindow.xaml.cs File Reference . . . . .	116
7.34 SpeexCompression.cs File Reference . . . . .	116
7.35 StateConverter.cs File Reference . . . . .	116
7.36 TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference . . . . .	116
7.37 TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference . . . . .	116
7.38 TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference . . . . .	116

# Chapter 1

## Namespace Index

### 1.1 Packages

Here are the packages with brief descriptions (if available):

<a href="#">GraphClass</a>	??
<a href="#">Hake_WPF</a>	??
<a href="#">Hake_WPF.AudioVideoManagers</a>	??
<a href="#">Hake_WPF.CallCenterReference</a>	??
<a href="#">Hake_WPF.Conversion</a>	??
<a href="#">Hake_WPF.Network</a>	??
<a href="#">Hake_WPF.Properties</a>	??
<a href="#">XamlGeneratedNamespace</a>	??





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Application	
Hake_WPF.App . . . . .	??
Hake_WPF.App . . . . .	??
Hake_WPF.App . . . . .	??
Hake_WPF.AudioVideoManagers.AudioVideoTransferManager . . . . .	??
Hake_WPF.Conversion.CompressionHelper . . . . .	??
Hake_WPF.Connection . . . . .	??
DuplexClientBase< Hake_WPF.CallCenterReference.IWcfCallCenterService >	
Hake_WPF.CallCenterReference.WcfCallCenterServiceClient . . . . .	??
GraphClass.Graph . . . . .	??
IClientChannel	
Hake_WPF.CallCenterReference.IWcfCallCenterServiceChannel . . . . .	??
IComponentConnector	
Hake_WPF.MainWindow . . . . .	??
Hake_WPF.MainWindow . . . . .	??
Hake_WPF.MapWindow . . . . .	??
Hake_WPF.MapWindow . . . . .	??
Hake_WPF.SettingsWindow . . . . .	??
Hake_WPF.SettingsWindow . . . . .	??
IExtensibleDataObject	
Hake_WPF.CallCenterReference.CallCenterConnectionDto . . . . .	??
Hake_WPF.CallCenterReference.ConnectionDto . . . . .	??
Hake_WPF.CallCenterReference.EmergencyTypeDto . . . . .	??
Hake_WPF.CallCenterReference.Fault . . . . .	??
Hake_WPF.CallCenterReference.ConnectionFault . . . . .	??
Hake_WPF.CallCenterReference.ParameterFault . . . . .	??
Hake_WPF.CallCenterReference.TargetStateFault . . . . .	??
Hake_WPF.CallCenterReference.LocationInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MeasurementInstrumentDto . . . . .	??
Hake_WPF.CallCenterReference.MediaConfigurationDto . . . . .	??
Hake_WPF.CallCenterReference.MediaInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MobileDeviceInformationDto . . . . .	??
Hake_WPF.CallCenterReference.PersonalInformationDto . . . . .	??
Hake_WPF.CallCenterReference.TextMessageDto . . . . .	??
Hake_WPF.CallCenterReference.UserCredentialsDto . . . . .	??
INotifyPropertyChanged	
Hake_WPF.Assignment . . . . .	??
Hake_WPF.CallCenterReference.CallCenterConnectionDto . . . . .	??

Hake_WPF.CallCenterReference.ConnectionDto . . . . .	??
Hake_WPF.CallCenterReference.EmergencyTypeDto . . . . .	??
Hake_WPF.CallCenterReference.Fault . . . . .	??
Hake_WPF.CallCenterReference.LocationInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MeasurementInstrumentDto . . . . .	??
Hake_WPF.CallCenterReference.MediaConfigurationDto . . . . .	??
Hake_WPF.CallCenterReference.MediaInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MobileDeviceInformationDto . . . . .	??
Hake_WPF.CallCenterReference.PersonalInformationDto . . . . .	??
Hake_WPF.CallCenterReference.TextMessageDto . . . . .	??
Hake_WPF.CallCenterReference.UserCredentialsDto . . . . .	??
InternalTypeHelper	
XamlGeneratedNamespace.GeneratedInternalTypeHelper . . . . .	??
XamlGeneratedNamespace.GeneratedInternalTypeHelper . . . . .	??
IValueConverter	
Hake_WPF.Conversion.AddressConverter . . . . .	??
Hake_WPF.Conversion.BoolConverter . . . . .	??
Hake_WPF.Conversion.LocationConverter . . . . .	??
Hake_WPF.Conversion.MinuteConverter . . . . .	??
Hake_WPF.Conversion.PhoneNumberConverter . . . . .	??
Hake_WPF.Conversion.PriorityConverter . . . . .	??
Hake_WPF.Conversion.StateConverter . . . . .	??
Hake_WPF.CallCenterReference.IWcfCallCenterService . . . . .	??
Hake_WPF.CallCenterReference.IWcfCallCenterServiceChannel . . . . .	??
Hake_WPF.CallCenterReference.WcfCallCenterServiceClient . . . . .	??
Hake_WPF.CallCenterReference.IWcfCallCenterServiceCallback . . . . .	??
Hake_WPF.AsyncReceiver . . . . .	??
ListBoxItem	
Hake_WPF.Assignment . . . . .	??
Hake_WPF.MapController . . . . .	??
object	
Hake_WPF.CallCenterReference.CallCenterConnectionDto . . . . .	??
Hake_WPF.CallCenterReference.ConnectionDto . . . . .	??
Hake_WPF.CallCenterReference.EmergencyTypeDto . . . . .	??
Hake_WPF.CallCenterReference.Fault . . . . .	??
Hake_WPF.CallCenterReference.LocationInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MeasurementInstrumentDto . . . . .	??
Hake_WPF.CallCenterReference.MediaConfigurationDto . . . . .	??
Hake_WPF.CallCenterReference.MediaInformationDto . . . . .	??
Hake_WPF.CallCenterReference.MobileDeviceInformationDto . . . . .	??
Hake_WPF.CallCenterReference.PersonalInformationDto . . . . .	??
Hake_WPF.CallCenterReference.TextMessageDto . . . . .	??
Hake_WPF.CallCenterReference.UserCredentialsDto . . . . .	??
Hake_WPF.Network.Pinger . . . . .	??
Hake_WPF.Settings . . . . .	??
Hake_WPF.AudioVideoManagers.SpeexCompression . . . . .	??
WaveStream	
Hake_WPF.AudioVideoManagers.BufferWaveStream . . . . .	??
Window	
Hake_WPF.SettingsWindow . . . . .	??
Window	
Hake_WPF.MainWindow . . . . .	??
Hake_WPF.MainWindow . . . . .	??
Hake_WPF.MainWindow . . . . .	??
Hake_WPF.MapWindow . . . . .	??
Hake_WPF.MapWindow . . . . .	??
Hake_WPF.MapWindow . . . . .	??
Hake_WPF.SettingsWindow . . . . .	??

Hake\_WPF.SettingsWindow . . . . . ??



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">Hake_WPF.Conversion.AddressConverter</a>	Converts PersonalInformationDto string that it (Streetname, postcode, locality). Converter is NOT implemented and returns always null! . . . . .	??
<a href="#">Hake_WPF.App</a>	Interaction logic for App.xaml . . . . .	??
<a href="#">Hake_WPF.Assignment</a>	Assignment listboxitem that contains many properties like handlers, location, time, state, priority, pushpins, assignment info and street name. It sets content and background depending state and priority and if there is location available. . . . .	??
<a href="#">Hake_WPF.AsyncReceiver</a>	Receives data from server which invokes event and writes it to debug. . . . .	??
<a href="#">Hake_WPF.AudioVideoManagers.AudioVideoTransferManager</a>	Class for processing incoming and outgoing audio and images. Handles media data received from the server. Reproduces speex encoded audio and Wave file segments using the primary audio output device in the system . . . . .	??
<a href="#">Hake_WPF.Conversion.BoolConverter</a>	Converts bool to string. True=on and false=ei. . . . .	??
<a href="#">Hake_WPF.AudioVideoManagers.BufferWaveStream</a>	Provides a NAudio WaveStream with infinite length to facilitate streaming audio playback. Contains an internal buffer for the audio samples. If the buffer is empty, all read requests will return the desired length of silence. If there is pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length . . . . .	??
<a href="#">Hake_WPF.CallCenterReference.CallCenterConnectionDto</a>		??
<a href="#">Hake_WPF.Conversion.CompressionHelper</a>	Class for static compression methods . . . . .	??
<a href="#">Hake_WPF.Connection</a>	Handles communication between server and this client. User must use connect method to connect server. After that user can use rest of the methods to manage connection. Also user should handle most of events, but atleast connectionupdatedevent. . . . .	??
<a href="#">Hake_WPF.CallCenterReference.ConnectionDto</a>		??
<a href="#">Hake_WPF.CallCenterReference.ConnectionFault</a>		??
<a href="#">Hake_WPF.CallCenterReference.EmergencyTypeDto</a>		??
<a href="#">Hake_WPF.CallCenterReference.Fault</a>		??
<a href="#">XamlGeneratedNamespace.GeneratedInternalTypeHelper</a>		
<a href="#">GeneratedInternalTypeHelper</a>		??
<a href="#">GraphClass.Graph</a>	Adds graph to given grid. user can add points, set bytespersecond and scroll to end. . . . .	??
<a href="#">Hake_WPF.CallCenterReference.IWcfCallCenterService</a>		??

<a href="#">Hake_WPF.CallCenterReference.IWcfCallCenterServiceCallback</a>	??
<a href="#">Hake_WPF.CallCenterReference.IWcfCallCenterServiceChannel</a>	??
<a href="#">Hake_WPF.Conversion.LocationConverter</a>	
Converts location to string	??
<a href="#">Hake_WPF.CallCenterReference.LocationInformationDto</a>	??
<a href="#">Hake_WPF.MainWindow</a>	
Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.	??
<a href="#">Hake_WPF.MapController</a>	
Controls maps. User can add specific pushpin to maps or all pushpins again. Maps can be too added. User can also center maps. Also handles closing maps.	??
<a href="#">Hake_WPF.MapWindow</a>	
Simple resizable window that only contains map.	??
<a href="#">Hake_WPF.CallCenterReference.MeasurementInstrumentDto</a>	??
<a href="#">Hake_WPF.CallCenterReference.MediaConfigurationDto</a>	??
<a href="#">Hake_WPF.CallCenterReference.MedialInformationDto</a>	??
<a href="#">Hake_WPF.Conversion.MinuteConverter</a>	
Class for converting time in minutes to a string and back.	??
<a href="#">Hake_WPF.CallCenterReference.MobileDeviceInformationDto</a>	??
<a href="#">Hake_WPF.CallCenterReference.ParameterFault</a>	??
<a href="#">Hake_WPF.CallCenterReference.PersonalInformationDto</a>	??
<a href="#">Hake_WPF.Conversion.PhoneNumberConverter</a>	
Converter for location.	??
<a href="#">Hake_WPF.Network.Pinger</a>	
Class for sending keepalive messages to server.	??
<a href="#">Hake_WPF.Conversion.PriorityConverter</a>	
Converter for priority to string.	??
<a href="#">Hake_WPF.Settings</a>	
User can store object that they specifie with key. User can add, update, get and remove those objects using the key.	??
<a href="#">Hake_WPF.SettingsWindow</a>	
SettingsWindow	??
<a href="#">Hake_WPF.AudioVideoManagers.SpeexCompression</a>	
Class providing static methods for compression and decompression of speex encoded audio segments.	??
<a href="#">Hake_WPF.Conversion.StateConverter</a>	
Converts state to string.	??
<a href="#">Hake_WPF.CallCenterReference.TargetStateFault</a>	??
<a href="#">Hake_WPF.CallCenterReference.TextMessageDto</a>	??
<a href="#">Hake_WPF.CallCenterReference.UserCredentialsDto</a>	??
<a href="#">Hake_WPF.CallCenterReference.WcfCallCenterServiceClient</a>	??

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AddressConverter.cs</a>	??
<a href="#">App.g.cs</a>	??
<a href="#">App.g.i.cs</a>	??
<a href="#">App.xaml.cs</a>	??
<a href="#">AssemblyInfo.cs</a>	??
<a href="#">Assignment.cs</a>	??
<a href="#">AudioVideoTransferManager.cs</a>	??
<a href="#">BoolConverter.cs</a>	??
<a href="#">BufferWaveStream.cs</a>	??
<a href="#">CompressionHelper.cs</a>	??
<a href="#">Connection.cs</a>	??
<a href="#">GeneratedInternalTypeHelper.g.cs</a>	??
<a href="#">GeneratedInternalTypeHelper.g.i.cs</a>	??
<a href="#">GraphClass.cs</a>	??
<a href="#">LocationConverter.cs</a>	??
<a href="#">MainWindow.g.cs</a>	??
<a href="#">MainWindow.g.i.cs</a>	??
<a href="#">MainWindow.xaml.cs</a>	??
<a href="#">MapController.cs</a>	??
<a href="#">MapWindow.g.cs</a>	??
<a href="#">MapWindow.g.i.cs</a>	??
<a href="#">MapWindow.xaml.cs</a>	??
<a href="#">MinuteConverter.cs</a>	??
<a href="#">PhoneNumberConverter.cs</a>	??
<a href="#">Pinger.cs</a>	??
<a href="#">PriorityConverter.cs</a>	??
<a href="#">Reference.cs</a>	??
<a href="#">Resources.Designer.cs</a>	??
<a href="#">Settings.cs</a>	??
<a href="#">Settings.Designer.cs</a>	??
<a href="#">SettingsWindow.g.cs</a>	??
<a href="#">SettingsWindow.g.i.cs</a>	??
<a href="#">SettingsWindow.xaml.cs</a>	??
<a href="#">SpeexCompression.cs</a>	??
<a href="#">StateConverter.cs</a>	??
<a href="#">TemporaryGeneratedFile_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs</a>	??
<a href="#">TemporaryGeneratedFile_5937a670-0e60-4077-877b-f7221da3dda1.cs</a>	??
<a href="#">TemporaryGeneratedFile_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs</a>	??





## Chapter 5

# Namespace Documentation

### 5.1 Package GraphClass

#### Classes

- class [Graph](#)

*Adds graph to given grid. user can add points, set bytespersecond and scroll to end.*

### 5.2 Package Hake\_WPF

#### Namespaces

- package [AudioVideoManagers](#)
- package [CallCenterReference](#)
- package [Conversion](#)
- package [Network](#)
- package [Properties](#)

#### Classes

- class [App](#)

*Interaction logic for App.xaml*

- class [Assignment](#)

*Assignment listboxitem that contains many properties like handlers, location, time, state, priority, pushpins, assignment info and street name. It sets content and background depending state and priority and if there is location available.*

- class [AsyncReceiver](#)

*Receives data from server which invokes event and writes it to debug.*

- class [Connection](#)

*Handles communication between server and this client. User must use connect method to connect server. After that user can use rest of the methods to manage connection. Also user should handle most of events, but atleast connectionupdatedevent.*

- class [MainWindow](#)

*Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.*

- class [MapController](#)

*Controls maps. User can add specific pushpin to maps or all pushpins again. Maps can be too added. User can also center maps. Also handles closing maps.*

- class [MapWindow](#)

*Simple resizable window that only contains map.*

- class [Settings](#)

*User can store object that they specify with key. User can add, update, get and remove those objects using the key.*

- class [SettingsWindow](#)

*SettingsWindow*

## Functions

- delegate void [ConnectionsUpdated](#) ([ConnectionDto](#)[] newConnections)

*Invoked when connection receives update.*

- delegate void [TcpMediaDataReceived](#) (object sender, [MediaInformationDto](#) info, byte[] data)

*Invoked when TcpMedia is received.*

- delegate void [UdpMediaDataReceived](#) (object sender, [MediaPacket](#) media)

*Invoked when udp media is received.*

- delegate void [MeasurementDataReceivedDelegate](#) (object sender, EventArgs e, [MeasurementInstrumentDto](#) instrument, byte[] measurementData, int dataSamplesPerSecond)

### 5.2.1 Function Documentation

#### 5.2.1.1 delegate void Hake\_WPF.ConnectionsUpdated ( [ConnectionDto](#)[] newConnections )

Invoked when connection receives update.

Parameters

<i>newConnections</i>	Connections as <a href="#">ConnectionDto</a>
-----------------------	--

#### 5.2.1.2 delegate void Hake\_WPF.MeasurementDataReceivedDelegate ( object sender, EventArgs e, [MeasurementInstrumentDto](#) instrument, byte[] measurementData, int dataSamplesPerSecond )

#### 5.2.1.3 delegate void Hake\_WPF.TcpMediaDataReceived ( object sender, [MediaInformationDto](#) info, byte[] data )

Invoked when TcpMedia is received.

Parameters

<i>sender</i>	Sender
<i>info</i>	Information
<i>data</i>	Data

#### 5.2.1.4 delegate void Hake\_WPF.UdpMediaDataReceived ( object sender, [MediaPacket](#) media )

Invoked when udp media is received.

Parameters

<i>sender</i>	Sender
<i>media</i>	Media

## 5.3 Package Hake\_WPF.AudioVideoManagers

### Classes

- class [AudioVideoTransferManager](#)  
*Class for processing incoming and outgoing audio and images. Handles media data received from the server. Reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.*
- class [BufferWaveStream](#)  
*Provides a NAudio WaveStream with infinite length to facilitate streaming audio playback. Contains an internal buffer for the audio samples. If the buffer is empty, all read requests will return the desired length of silence. If there is pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length.*
- class [SpeexCompression](#)  
*Class providing static methods for compression and decompression of speex encoded audio segments.*

### Functions

- delegate void [JpgImageReceived](#) (object sender, byte[] imageData)  
*Delegate for image receive events.*

#### 5.3.1 Function Documentation

##### 5.3.1.1 delegate void Hake\_WPF.AudioVideoManagers.JpgImageReceived ( object sender, byte[] imageData )

Delegate for image receive events.

#### Parameters

<i>sender</i>	The publishing <a href="#">AudioVideoTransferManager</a> instance
<i>imageData</i>	Byte representation of a jpg compressed image

## 5.4 Package Hake\_WPF.CallCenterReference

### Classes

- class [CallCenterConnectionDto](#)
- class [ConnectionDto](#)
- class [ConnectionFault](#)
- class [EmergencyTypeDto](#)
- class [Fault](#)
- interface [IWcfCallCenterService](#)
- interface [IWcfCallCenterServiceCallback](#)
- interface [IWcfCallCenterServiceChannel](#)
- class [LocationInformationDto](#)
- class [MeasurementInstrumentDto](#)
- class [MediaConfigurationDto](#)
- class [MediaInformationDto](#)
- class [MobileDeviceInformationDto](#)
- class [ParameterFault](#)
- class [PersonalInformationDto](#)
- class [TargetStateFault](#)
- class [TextMessageDto](#)
- class [UserCredentialsDto](#)
- class [WcfCallCenterServiceClient](#)

## Enumerations

- enum [ConnectionPriorityDto](#) : int { [ConnectionPriorityDto.NotUrgent](#) = 0, [ConnectionPriorityDto.Urgent](#) = 1 }
- enum [ConnectionStateDto](#) : int { [ConnectionStateDto.Arrived](#) = 0, [ConnectionStateDto.InProgress](#) = 1, [ConnectionStateDto.InTransfer](#) = 2, [ConnectionStateDto.Processed](#) = 3, [ConnectionStateDto.Hold](#) = 4 }
- enum [LocationTypeDto](#) : int { [LocationTypeDto.Initial](#) = 0, [LocationTypeDto.Response](#) = 1, [LocationTypeDto.Movement](#) = 2, [LocationTypeDto.UserSpecified](#) = 3 }
- enum [MeasurementInstrumentTypeDto](#) : int { [MeasurementInstrumentTypeDto.ECG](#) = 0 }
- enum [RemoteActionDto](#) : int { [RemoteActionDto.requestLocation](#) = 0, [RemoteActionDto.requestDeviceStatusinformation](#) = 1, [RemoteActionDto.requestShowLocationMap](#) = 2, [RemoteActionDto.requestInstrumentList](#) = 3 }
- enum [MediaTypeDto](#) : int { [MediaTypeDto.Wave](#) = 0, [MediaTypeDto.Jpeg](#) = 1, [MediaTypeDto.AAC](#) = 2, [MediaTypeDto.H264](#) = 3, [MediaTypeDto.MP4](#) = 4, [MediaTypeDto.Opus](#) = 5, [MediaTypeDto.Speex](#) = 6 }

### 5.4.1 Enumeration Type Documentation

5.4.1.1 enum [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto](#) : int

Enumerator

***NotUrgent***

***Urgent***

5.4.1.2 enum [Hake\\_WPF.CallCenterReference.ConnectionStateDto](#) : int

Enumerator

***Arrived***

***InProgress***

***InTransfer***

***Processed***

***Hold***

5.4.1.3 enum [Hake\\_WPF.CallCenterReference.LocationTypeDto](#) : int

Enumerator

***Initial***

***Response***

***Movement***

***UserSpecified***

5.4.1.4 enum [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto](#) : int

Enumerator

***ECG***

## 5.4.1.5 enum Hake\_WPF.CallCenterReference.MediaTypeDto : int

Enumerator

**Wave**  
**Jpeg**  
**AAC**  
**H264**  
**MP4**  
**Opus**  
**Speex**

## 5.4.1.6 enum Hake\_WPF.CallCenterReference.RemoteActionDto : int

Enumerator

**requestLocation**  
**requestDeviceStatusinformation**  
**requestShowLocationMap**  
**requestInstrumentList**

## 5.5 Package Hake\_WPF.Conversion

### Classes

- class [AddressConverter](#)  
*Converts PersonalInformationDto string that it (Streetname, postalcode, locality). Converback is NOT implemented and returns always null!*
- class [BoolConverter](#)  
*Converts bool to string. True=on and false=ei.*
- class [CompressionHelper](#)  
*Class for static compression methods.*
- class [LocationConverter](#)  
*Converts location to string*
- class [MinuteConverter](#)  
*Class for converting time in minutes to a string and back.*
- class [PhoneNumberConverter](#)  
*Converter for location.*
- class [PriorityConverter](#)  
*Converter for priority to string.*
- class [StateConverter](#)  
*Converts state to string.*

## 5.6 Package Hake\_WPF.Network

### Classes

- class [Pinger](#)  
*Class for sending keepalive messages to server.*

## 5.7 Package Hake\_WPF.Properties

### Classes

- class **Resources**  
*A strongly-typed resource class, for looking up localized strings, etc.*
- class **Settings**

## 5.8 Package XamlGeneratedNamespace

### Classes

- class [GeneratedInternalTypeHelper](#)  
*GeneratedInternalTypeHelper*

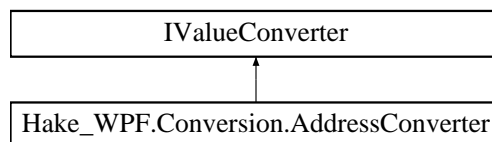
## Chapter 6

# Class Documentation

### 6.1 Hake\_WPF.Conversion.AddressConverter Class Reference

Converts PersonallnformationDto string that it (Streetname, postalcode, locality). Converback is NOT implemented and returns always null!

Inheritance diagram for Hake\_WPF.Conversion.AddressConverter:



#### Public Member Functions

- object **Convert** (object value, Type targetType, object parameter, CultureInfo culture)  
*Tries cast value to PersonallnformationDto and then returns string that is compination of streetname, postalcode and locality.*
- object **ConvertBack** (object value, Type targetType, object parameter, CultureInfo culture)  
*returns always null.*

#### 6.1.1 Detailed Description

Converts PersonallnformationDto string that it (Streetname, postalcode, locality). Converback is NOT implemented and returns always null!

<author>Atte Söderlund</author>

#### 6.1.2 Member Function Documentation

##### 6.1.2.1 object Hake\_WPF.Conversion.AddressConverter.Convert ( object value, Type targetType, object parameter, CultureInfo culture )

Tries cast value to PersonallnformationDto and then returns string that is compination of streetname, postalcode and locality.

**Parameters**

<i>value</i>	Given value that should be type of PersonalInformationDto
<i>targetType</i>	This is ignored in code so do not use this
<i>parameter</i>	Parameter are not in use so do not use this
<i>culture</i>	Culture is not in use so do not use this

**Returns**

Address as string

6.1.2.2 object Hake\_WPF.Conversion.AddressConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

returns always null.

**Returns**

Null

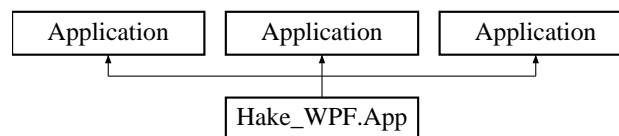
The documentation for this class was generated from the following file:

- [AddressConverter.cs](#)

## 6.2 Hake\_WPF.App Class Reference

Interaction logic for App.xaml

Inheritance diagram for Hake\_WPF.App:

**Public Member Functions**

- void [InitializeComponent](#) ()  
*InitializeComponent*
- void [InitializeComponent](#) ()  
*InitializeComponent*

**Static Public Member Functions**

- static void [Main](#) ()  
*Application Entry Point.*
- static void [Main](#) ()  
*Application Entry Point.*



### 6.2.1 Detailed Description

Interaction logic for App.xaml

[App](#)

### 6.2.2 Member Function Documentation

#### 6.2.2.1 void Hake\_WPF.App.InitializeComponent ( )

InitializeComponent

#### 6.2.2.2 void Hake\_WPF.App.InitializeComponent ( )

InitializeComponent

#### 6.2.2.3 static void Hake\_WPF.App.Main ( ) [static]

Application Entry Point.

#### 6.2.2.4 static void Hake\_WPF.App.Main ( ) [static]

Application Entry Point.

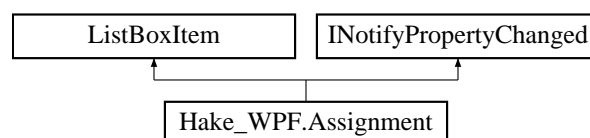
The documentation for this class was generated from the following files:

- [App.xaml.cs](#)
- [App.g.cs](#)
- [App.g.i.cs](#)

## 6.3 Hake\_WPF.Assignment Class Reference

Assignment listboxitem that contains many properties like handlers, location, time, state, priority, pushpins, assignment info and street name. It sets content and background depending state and priority and if there is location available.

Inheritance diagram for Hake\_WPF.Assignment:



### Public Types

- enum [priorities](#) { [priorities.NotUrgent](#) = 0, [priorities.Urgent](#) = 1 }
- Not urgent and urgent enums. NotUrgent is 0 and Urgent is 1 as ints.
- enum [States](#) { [States.New](#) = 0, [States.InProgress](#) = 1, [States.InTransfer](#) = 2, [States.Finalized](#) = 3, [States.Hold](#) = 4 }
- Enum of states. New = 0,...,Hold=4.

## Public Member Functions

- [Assignment](#) (String [guid](#), DateTime [time](#), [priorities](#) [priority](#), [States](#) [state](#), [LocationInformationDto](#) [location](#))  
*Initializes Assignment: sets state, priority, time, location, assimentinfo, sets backgroundcolor and content. These are essential for assignment*
- [Assignment](#) (String [guid](#), DateTime [time](#), [priorities](#) [priority](#), [States](#) [state](#))  
*Constructor without location.*

## Static Public Attributes

- static readonly DependencyProperty [IsHandlerProperty](#)  
*DependencyProperty for ishandler.*

## Properties

- String [EmergencyType](#) [get, set]  
*Sets and getsEmergency type can be any string that is short description about emergency.*
- ObservableCollection  
< [TextMessageDto](#) > [TextMessages](#) [get, set]  
*Textmessages in observablecollection.*
- [PersonalInformationDto](#) [PersonalInfo](#) [get, set]  
*Sets and gets personal information as PersonalInformationDto that includes name, address and list of phonenumbers and all as strings.*
- [priorities](#) [Priority](#) [get, set]  
*Sets and gets priority of this assignment. When value is set, also sets background and content.*
- String [Guid](#) [get, set]  
*Sets and gets guid.*
- DateTime [Time](#) [get, set]  
*Sets and gets time. This time is when assignment is taken in. When setted, also sets background color and content.*
- bool [NoSound](#) [get, set]  
*Gets and sets NoSound.*
- [States](#) [State](#) [get, set]  
*Sets value and also takes care of background color and listboxitem content. Also if state is finalized removes pushpins.*
- bool [IsHandler](#) [get, set]  
*Sets and gets IsHandler*
- [LocationInformationDto](#) [Location](#) [get, set]  
*Sets and gets location as LocationInformationDto that contains location, accuracy in meters and time. When this is setted, sets content also.*
- double [LocationAccuracyMeters](#) [get, set]  
*Sets and gets locationAccuracyMeters*
- DateTimeOffset [LocationAcquisitionTime](#) [get, set]  
*Sets and gets LocationAcquisitionTime.*
- ObservableCollection< Pushpin > [Pushpins](#) [get, set]  
*Sets and gets Pushpins as ObservableCollention. If assignemnts state is finalized, it won't set pushpins. Also adds pushpins in index 1 background to light green.*
- String [StreetName](#) [get, set]  
*Sets and gets streetname. When setted, also sets content and invokes NotifyPropertyChanged.*
- [MobileDeviceInformationDto](#) [DeviceInfo](#) [get, set]  
*Sets and gets DeviceInfo.*

## Events

- PropertyChangedEventHandler [PropertyChanged](#)

## Private Member Functions

- void [NotifyPropertyChanged](#) (String info)  
*Notify that property is changed. This will update bindings.*
- void [SetContent](#) ()  
*Sets different content for listboxitem depending on what information is already set. For location uses format 00.0000 that shows 4 decimals of latitude and longitude. Also converts other text to right form like state and priority.*
- void [SetBackgroundColor](#) ()  
*Depending on state and priority, sets backgroundcolor for listboxitem and for pushpin.*
- void [getStreetNameFromLocationAsync](#) ()  
*Updates street name according to location with Bing Maps API Function makes asynchronous call to Bing Maps API rest service which returns information about given location*
- void [DownloadStringCallback](#) (Object sender, DownloadStringCompletedEventArgs e)  
*Callback function for WebClient Download completed event. This function sets Streetname according to location.*

## Private Attributes

- WebClient [WebClientForLocationQuery](#)
- String [emergencyType](#)
- ObservableCollection  
    < [TextMessageDto](#) > [textMessages](#) = new ObservableCollection<[TextMessageDto](#)>()
- [PersonallInformationDto](#) [personallInfo](#)
- [priorities](#) [priority](#)
- String [guid](#) = ""
- DateTime [time](#)
- bool [noSound](#)
- [States](#) [state](#)
- List< Brush > [stateBrushes](#)
- [LocationInformationDto](#) [location](#)
- double [locationAccuracyMeters](#)
- DateTimeOffset [locationAcquisitionTime](#)
- ObservableCollection< Pushpin > [pushpins](#) = new ObservableCollection<Pushpin>()
- String [streetName](#)
- [MobileDeviceInformationDto](#) [deviceInfo](#)

### 6.3.1 Detailed Description

Assignment listboxitem that contains many properties like handlers, location, time, state, priority, pushpins, assignment info and street name. It sets content and background depending state and priority and if there is location available.

<author>Atte Söderlund</author>

### 6.3.2 Member Enumeration Documentation

#### 6.3.2.1 enum Hake\_WPF.Assignment.priorities

Not urgent and urgent enums. NotUrgent is 0 and Urgent is 1 as ints.

Enumerator

***NotUrgent***

***Urgent***

#### 6.3.2.2 enum Hake\_WPF.Assignment.States

Enum of states. New = 0,...,Hold=4.

Enumerator

***New***

***InProgress***

***InTransfer***

***Finalized***

***Hold***

### 6.3.3 Constructor & Destructor Documentation

#### 6.3.3.1 Hake\_WPF.Assignment.Assignment ( String *guid*, DateTime *time*, priorities *priority*, States *state*, LocationInformationDto *location* )

Initializes Assignment: sets state, priority, time, location, assimentinfo, sets backgroundcolor and content. These are essential for assignment

Parameters

<i>time</i>	Assignment connection time
<i>priority</i>	Assignments priority from Assignment.Priorities
<i>state</i>	Assignments state from Assignment.States
<i>location</i>	Assignments location

#### 6.3.3.2 Hake\_WPF.Assignment.Assignment ( String *guid*, DateTime *time*, priorities *priority*, States *state* )

Constructor without location.

Parameters

<i>time</i>	Assignment connection time
<i>priority</i>	Assignments priority from Assignment.Priorities
<i>state</i>	Assignments state from Assignment.States

### 6.3.4 Member Function Documentation

#### 6.3.4.1 void Hake\_WPF.Assignment.DownloadStringCallback ( Object *sender*, DownloadStringCompletedEventArgs *e* ) [private]

Callback function for WebClient Download completed event. This function sets Streetname according to location.

<author>Niko Mononen</author>

## Parameters

<i>sender</i>	Callback sender
<i>e</i>	Download completed event arguments

## 6.3.4.2 void Hake\_WPF.Assignment.getStreetNameFromLocationAsync ( ) [private]

Updates street name according to location with Bing Maps API Function makes asynchronous call to Bing Maps API rest service which returns information about given location

<author>Niko Mononen</author>

## 6.3.4.3 void Hake\_WPF.Assignment.NotifyPropertyChanged ( String info ) [private]

Notify that property is changed. This will update bindings.

## Parameters

<i>info</i>	Info string
-------------	-------------

## 6.3.4.4 void Hake\_WPF.Assignment.SetBackgroundColor ( ) [private]

Depending on state and priority, sets backgroundcolor for listboxitem and for pushpin.

## 6.3.4.5 void Hake\_WPF.Assignment.SetContent ( ) [private]

Sets different content for listboxitem depending on what information is already set. For location uses format 00.0000 that shows 4 decimals of latitude and longitude. Also converts other text to right form like state and priority.

## 6.3.5 Member Data Documentation

## 6.3.5.1 MobileDeviceInformationDto Hake\_WPF.Assignment.deviceInfo [private]

## 6.3.5.2 String Hake\_WPF.Assignment.emergencyType [private]

## 6.3.5.3 String Hake\_WPF.Assignment.guid = "" [private]

## 6.3.5.4 readonly DependencyProperty Hake\_WPF.Assignment.IsHandlerProperty [static]

## Initial value:

```
=
    DependencyProperty.Register("IsHandler", typeof(bool), typeof(MainWindow), new PropertyMetadata(
        (false)))
```

DependencyProperty for ishandler.

## 6.3.5.5 LocationInformationDto Hake\_WPF.Assignment.location [private]

## 6.3.5.6 double Hake\_WPF.Assignment.locationAccuracyMeters [private]

## 6.3.5.7 DateTimeOffset Hake\_WPF.Assignment.locationAcquisitionTime [private]

6.3.5.8 `bool Hake_WPF.Assignment.noSound` [private]

6.3.5.9 `PersonalInformationDto Hake_WPF.Assignment.personalInfo` [private]

6.3.5.10 `priorities Hake_WPF.Assignment.priority` [private]

6.3.5.11 `ObservableCollection<Pushpin> Hake_WPF.Assignment.pushpins = new ObservableCollection<Pushpin>()` [private]

6.3.5.12 `States Hake_WPF.Assignment.state` [private]

6.3.5.13 `List<Brush> Hake_WPF.Assignment.stateBrushes` [private]

#### Initial value:

```
= new List<Brush>
{
    new SolidColorBrush(new Color() { B = 48, R = 230, G = 48, A = 255 }),
    new SolidColorBrush(new Color() { B = 235, R = 89, G = 63, A = 255 }),
    new SolidColorBrush(new Color() { B = 76, R = 252, G = 238, A = 255 }),
    new SolidColorBrush(new Color() { B = 131, R = 133, G = 132, A = 255 }),
    new SolidColorBrush(new Color() { B = 79, R = 247, G = 205, A = 255 })
}
```

6.3.5.14 `String Hake_WPF.Assignment.streetName` [private]

6.3.5.15 `ObservableCollection<TextMessageDto> Hake_WPF.Assignment.textMessages = new ObservableCollection<TextMessageDto>()` [private]

6.3.5.16 `DateTime Hake_WPF.Assignment.time` [private]

6.3.5.17 `WebClient Hake_WPF.Assignment.WebClientForLocationQuery` [private]

### 6.3.6 Property Documentation

6.3.6.1 `MobileDeviceInformationDto Hake_WPF.Assignment.DeviceInfo` [get], [set]

Sets and gets DeviceInfo.

6.3.6.2 `String Hake_WPF.Assignment.EmergencyType` [get], [set]

Sets and getsEmergency type can be any string that is short description about emergency.

6.3.6.3 `String Hake_WPF.Assignment.Guid` [get], [set]

Sets and gets guid.

6.3.6.4 `bool Hake_WPF.Assignment.IsHandler` [get], [set]

Sets and gets IsHandler

6.3.6.5 `LocationInformationDto Hake_WPF.Assignment.Location` [get], [set]

Sets and gets location as LocationInformationDto that contains location, accuracy in meters and time. When this is setted, sets content also.

**6.3.6.6** `double Hake_WPF.Assignment.LocationAccuracyMeters` `[get]`, `[set]`

Sets and gets locationAccuracyMeters

**6.3.6.7** `DateTimeOffset Hake_WPF.Assignment.LocationAcquisitionTime` `[get]`, `[set]`

Sets and gets LocationAcquisitionTime.

**6.3.6.8** `bool Hake_WPF.Assignment.NoSound` `[get]`, `[set]`

Gets and sets NoSound.

**6.3.6.9** `PersonalInformationDto Hake_WPF.Assignment.PersonalInfo` `[get]`, `[set]`

Sets and gets personal information as PersonalInformationDto that includes name, address and list of phonenumbers and all as strings.

**6.3.6.10** `priorities Hake_WPF.Assignment.Priority` `[get]`, `[set]`

Sets and gets priority of this assignment. When value is set, also sets background and content.

**6.3.6.11** `ObservableCollection<Pushpin> Hake_WPF.Assignment.Pushpins` `[get]`, `[set]`

Sets and gets Pushpins as ObservableCollection. If assignments state is finalized, it won't set pushpins. Also adds pushpins in index 1 background to light green.

**6.3.6.12** `States Hake_WPF.Assignment.State` `[get]`, `[set]`

Sets value and also takes care of background color and listboxitem content. Also if state is finalized removes pushpins.

**6.3.6.13** `String Hake_WPF.Assignment.StreetName` `[get]`, `[set]`

Sets and gets streetname. When setted, also sets content and invokes NotifyPropertyChanged.

**6.3.6.14** `ObservableCollection<TextMessageDto> Hake_WPF.Assignment.TextMessages` `[get]`, `[set]`

Textmessages in observablecollection.

**6.3.6.15** `DateTime Hake_WPF.Assignment.Time` `[get]`, `[set]`

Sets and gets time. This time is when assignment is taken in. When setted, also sets background color and content.

## 6.3.7 Event Documentation

**6.3.7.1** `PropertyChangedEventHandler Hake_WPF.Assignment.PropertyChanged`

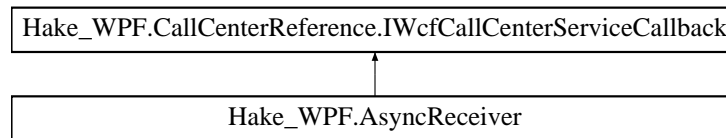
The documentation for this class was generated from the following file:

- [Assignment.cs](#)

## 6.4 Hake\_WPF.AsyncReceiver Class Reference

Receives data from server which invokes event and writes it to debug.

Inheritance diagram for Hake\_WPF.AsyncReceiver:



### Public Member Functions

- void [ActiveConnectionsUpdated](#) ([ConnectionDto](#)[] updatedConnections)  
*Is invoked everytime client receives update for assignment from server. This method only invokes [ConnectionsUpdatedEvent](#) so [Connection](#) class can handle it.*
- void [AudioVideoReceived](#) (string sourceGuid, [MediaInformationDto](#) mediaInfo, byte[] mediaData)  
*Invokes [tcpMediaDataReceivedEvent](#).*
- void [MeasurementDataReceived](#) (string sourceGuid, [MeasurementInstrumentDto](#) instrument, byte[] measurementData)  
*Invokes [MeasurementDateReceivedEvent](#).*

### Public Attributes

- [TcpMediaDataReceived](#) [TcpMediaDataReceivedEvent](#)
- [MeasurementDataReceivedDelegate](#) [MeasurementDataReceivedEvent](#)

### Events

- [ConnectionsUpdated](#) [ConnectionsUpdatedEvent](#)

#### 6.4.1 Detailed Description

Receives data from server which invokes event and writes it to debug.

#### 6.4.2 Member Function Documentation

##### 6.4.2.1 void Hake\_WPF.AsyncReceiver.ActiveConnectionsUpdated ( [ConnectionDto](#)[] *updatedConnections* )

Is invoked everytime client receives update for assignment from server. This method only invokes [ConnectionsUpdatedEvent](#) so [Connection](#) class can handle it.

<author>Veli-Mikko Puupponen</author>

#### Parameters

<i>updatedConnections</i>	
---------------------------	--



6.4.2.2 void Hake\_WPF.AsyncReceiver.AudioVideoReceived ( string *sourceGuid*, *MediaInformationDto* *mediaInfo*, byte[] *mediaData* )

Invokes tcpMediaDataReceivedEvent.

6.4.2.3 void Hake\_WPF.AsyncReceiver.MeasurementDataReceived ( string *sourceGuid*, *MeasurementInstrumentDto* *instrument*, byte[] *measurementData* )

Invokes MeasurementDateReceivedEvent.

### 6.4.3 Member Data Documentation

6.4.3.1 *MeasurementDataReceivedDelegate* Hake\_WPF.AsyncReceiver.MeasurementDataReceivedEvent

6.4.3.2 *TcpMediaDataReceived* Hake\_WPF.AsyncReceiver.TcpMediaDataReceivedEvent

### 6.4.4 Event Documentation

6.4.4.1 *ConnectionsUpdated* Hake\_WPF.AsyncReceiver.ConnectionsUpdatedEvent

The documentation for this class was generated from the following file:

- [Connection.cs](#)

## 6.5 Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager Class Reference

Class for processing incoming and outgoing audio and images. Handles media data received from the server. Reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.

### Public Member Functions

- [AudioVideoTransferManager](#) ([Connection](#) c)  
*Initializes a new [AudioVideoTransferManager](#) that uses the provided connection to receive and transmit audio and pictures.*
- void [EnableOutgoingAudio](#) ()  
*Enables audio recording and publishing to the server from the primary audio capture in the system.*
- void [DisableOutgoingAudio](#) ()  
*Stops recording audio and publishing it to the server.*
- void [EnableIncomingAudio](#) ()  
*Starts receiving audio from the server and reproducing it using the primary audio output device in the system.*
- void [DisableIncomingAudio](#) ()  
*Stops receiving and reproducing audio.*

### Public Attributes

- [JpgImageReceived](#) [JpgImageReceivedEvent](#)

### Properties

- bool [UseOutgoingUdp](#) [get, set]  
*Gets and sets UseOutGoingUdp.*

## Private Member Functions

- void [InitializeSpeex](#) ()  
*Initializes speex audio encoder and decoder.*
- void [InitializeAudioIODevices](#) ()  
*Initializes NAudio WaveIn and WaveOut used to capture and reproduce audio.*
- void [SpeexAudioReceived](#) (byte[] speexData, int originatingLength)  
*Handles incoming speex audio segments. Decompresses them into PCM that is added into the bufferedWaveStream from which the active WaveOut instance reads and reproduces them.*
- void [MicrophoneDataAvailable](#) (object sender, WaveInEventArgs e)  
*Handles PCM data segments captured by the active WaveIn instance.*
- void [UdpMediaReceivedHandler](#) (object sender, MediaPacket mediaPacket)  
*Handles incoming media received by the active [Connection](#) instance over the UDP channel. Supported formats are jpg images and speex encoded audio.*
- void [TcpMediaReceivedHandler](#) (object sender, [MediaInformationDto](#) info, byte[] data)  
*Handles incoming media received by the active [Connection](#) instance over the WCF TCP channel. Supported formats are jpg images and wave audio fragments with pcm payload.*
- void [PlayWaveFragment](#) (byte[] audio)  
*Plays a wave file using a new SoundPlayer instance.*

## Private Attributes

- [MediaInformation](#) [outgoingSpeexInformation](#) = new [MediaInformation](#)([MediaFormat.Speex](#), [Transfer↔](#) [Compression.None](#))
- [WaveFormat](#) [AudioCaptureFormat](#)
- const int [speexFramesInSecond](#) = 50
- const int [defaultSpeexQuality](#) = 2
- [SpeexDecoder](#) [decoder](#)
- [SpeexEncoder](#) [encoder](#)
- [IWaveIn](#) [recorder](#)
- [WaveOut](#) [player](#)
- bool [audioPlaybackActive](#) = false
- [BufferWaveStream](#) [bufferedWaveStream](#)
- object [playbackConfigurationLock](#) = new object()
- [Connection](#) [connection](#)
- bool [useOutgoingUdp](#) = true

### 6.5.1 Detailed Description

Class for processing incoming and outgoing audio and images. Handles media data received from the server. Reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.

<author>Veli-Mikko Puupponen</author> Captures audio from the default audio input device in the system using a NAudio WaveIn instance. If outgoing audio is enabled, uploads the captured audio in a speex encoded format to the server.

Publishes a [JpgImageReceivedEvent](#) upon receiving an image from the server.

### 6.5.2 Constructor & Destructor Documentation

#### 6.5.2.1 [Hake\\_WPF.AudioVideoManagers.AudioVideoTransferManager.AudioVideoTransferManager](#) ( [Connection c](#) )

Initializes a new [AudioVideoTransferManager](#) that uses the provided connection to receive and transmit audio and pictures.

## Parameters

<i>c</i>	Valid <a href="#">Connection</a> instance.
----------	--

## 6.5.3 Member Function Documentation

## 6.5.3.1 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.DisableIncomingAudio ( )

Stops receiving and reproducing audio.

## 6.5.3.2 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.DisableOutgoingAudio ( )

Stops recording audio and publishing it to the server.

## 6.5.3.3 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.EnableIncomingAudio ( )

Starts receiving audio from the server and reproducing it using the primary audio output device in the system.

## 6.5.3.4 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.EnableOutgoingAudio ( )

Enables audio recording and publishing to the server from the primary audio capture in the system.

## 6.5.3.5 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.InitializeAudioIODevices ( ) [private]

Initializes NAudio WaveIn and WaveOut used to capture and reproduce audio.

## 6.5.3.6 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.InitializeSpeex ( ) [private]

Initializes speex audio encoder and decoder.

6.5.3.7 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.MicrophoneDataAvailable ( object *sender*, WaveInEventArgs *e* ) [private]

Handles PCM data segments captured by the active WaveIn instance.

## Parameters

<i>sender</i>	The publishing WaveIn instance
<i>e</i>	WaveInEventArgs containing the captured audio

6.5.3.8 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.PlayWaveFragment ( byte[] *audio* ) [private]

Plays a wave file using a new SoundPlayer instance.

## Parameters

<i>audio</i>	Audio in wave format
--------------	----------------------

6.5.3.9 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.SpeexAudioReceived ( byte[] *speexData*, int *originatingLength* ) [private]

Handles incoming speex audio segments. Decompresses them into PCM that is added into the bufferedWave↔Stream from which the active WaveOut instance reads and reproduces them.

## Parameters

<i>speexData</i>	Speex encoded audio
<i>originatingLength</i>	The count of 16 bit PCM samples that were encoded to produce the provided encoded data

6.5.3.10 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.TcpMediaReceivedHandler ( object *sender*, **MediaInformationDto** *info*, byte[] *data* ) [private]

Handles incoming media received by the active [Connection](#) instance over the WCF TCP channel. Supported formats are jpg images and wave audio fragments with pcm payload.

## Parameters

<i>sender</i>	<a href="#">Connection</a> instance publishing the event
<i>info</i>	MediaInformationDto describing the media data
<i>data</i>	Media data payload

6.5.3.11 void Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.UdpMediaReceivedHandler ( object *sender*, **MediaPacket** *mediaPacket* ) [private]

Handles incoming media received by the active [Connection](#) instance over the UDP channel. Supported formats are jpg images and speex encoded audio.

## Parameters

<i>sender</i>	<a href="#">Connection</a> instance publishing the event
<i>mediaPacket</i>	MediaPacket containing the media data and description

## 6.5.4 Member Data Documentation

6.5.4.1 WaveFormat Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.AudioCaptureFormat [private]

6.5.4.2 bool Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.audioPlaybackActive = false [private]

6.5.4.3 BufferWaveStream Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.bufferedWaveStream [private]

6.5.4.4 Connection Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.connection [private]

6.5.4.5 SpeexDecoder Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.decoder [private]

6.5.4.6 const int Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.defaultSpeexQuality = 2 [private]

6.5.4.7 SpeexEncoder Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.encoder [private]

6.5.4.8 JpgImageReceived Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.JpgImageReceivedEvent

6.5.4.9 MediaInformation Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.outgoingSpeexInformation = new MediaInformation(MediaFormat.Speex, TransferCompression.None) [private]

6.5.4.10 object Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.playbackConfigurationLock = new object() [private]

6.5.4.11 WaveOut Hake\_WPF.AudioVideoManagers.AudioVideoTransferManager.player [private]

6.5.4.12 `IWaveIn Hake_WPF.AudioVideoManagers.AudioVideoTransferManager.recorder` [private]

6.5.4.13 `const int Hake_WPF.AudioVideoManagers.AudioVideoTransferManager.speexFramesInSecond = 50` [private]

6.5.4.14 `bool Hake_WPF.AudioVideoManagers.AudioVideoTransferManager.useOutgoingUdp = true` [private]

## 6.5.5 Property Documentation

6.5.5.1 `bool Hake_WPF.AudioVideoManagers.AudioVideoTransferManager.UseOutgoingUdp` [get],[set]

Gets and sets UseOutGoingUdp.

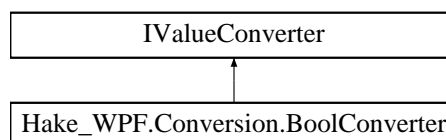
The documentation for this class was generated from the following file:

- [AudioVideoTransferManager.cs](#)

## 6.6 Hake\_WPF.Conversion.BoolConverter Class Reference

Converts bool to string. True=on and false=ei.

Inheritance diagram for Hake\_WPF.Conversion.BoolConverter:



### Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts bool to string*
- object [ConvertBack](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts String to bool.*

### 6.6.1 Detailed Description

Converts bool to string. True=on and false=ei.

<author>Atte Söderlund</author>

### 6.6.2 Member Function Documentation

6.6.2.1 `object Hake_WPF.Conversion.BoolConverter.Convert ( object value, Type targetType, object parameter, CultureInfo culture )`

Converts bool to string

Parameters

---

<i>value</i>	Bool that needs converting to string
<i>targetType</i>	Is not in use
<i>parameter</i>	Is not in use
<i>culture</i>	Is not in use

**Returns**

Bool as String

**6.6.2.2** object Hake\_WPF.Conversion.BoolConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

Converts String to bool.

**Parameters**

<i>value</i>	"true" or "false" strings
<i>targetType</i>	Is not in use
<i>parameter</i>	Is not in use
<i>culture</i>	Is not in use

**Returns**

Bool if can parse it from string

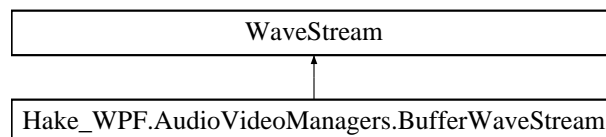
The documentation for this class was generated from the following file:

- [BoolConverter.cs](#)

## 6.7 Hake\_WPF.AudioVideoManagers.BufferWaveStream Class Reference

Provides a NAudio WaveStream with infinite length to facilitate streaming audio playback. Contains an internal buffer for the audio samples. If the buffer is empty, all read requests will return the desired length of silence. If there is pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length.

Inheritance diagram for Hake\_WPF.AudioVideoManagers.BufferWaveStream:

**Public Member Functions**

- [BufferWaveStream](#) (int sampleRate, int bytesPerSample, int channels)  
*Initializes a new [BufferWaveStream](#) that has a WaveFormat defined by the provided parameters. The audio buffer is empty and has no length limit.*
- override void [Write](#) (byte[] buffer, int offset, int count)  
*Writes the provided audio data to the outgoing PCM segment buffer.*
- override int [Read](#) (byte[] buffer, int offset, int count)  
*Reads PCM samples from the underlying buffer. If no data is available, returns silent samples.*

## Properties

- override WaveFormat [WaveFormat](#) [get]
- override long [Length](#) [get]
- override long [Position](#) [get, set]

## Private Attributes

- Queue< byte[]> [completePcmSegments](#)
- byte[] [overflowFromLastSegment](#)
- readonly WaveFormat [waveFormat](#)
- object [queueLock](#)
- long [lengthInSamples](#) = 1
- long [defaultPosition](#) = 0

### 6.7.1 Detailed Description

Provides a NAudio WaveStream with infinite length to facilitate streaming audio playback. Contains an internal buffer for the audio samples. If the buffer is empty, all read requests will return the desired length of silence. If there is pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length.

<author>Veli-Mikko Puupponen</author>

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 Hake\_WPF.AudioVideoManagers.BufferWaveStream.BufferWaveStream ( int *sampleRate*, int *bytesPerSample*, int *channels* )

Initializes a new [BufferWaveStream](#) that has a WaveFormat defined by the provided parameters. The audio buffer is empty and has no length limit.

##### Parameters

<i>sampleRate</i>	Sample rate in samples per second
<i>bytesPerSample</i>	bytes per PCM sample
<i>channels</i>	Number of channels

### 6.7.3 Member Function Documentation

#### 6.7.3.1 override int Hake\_WPF.AudioVideoManagers.BufferWaveStream.Read ( byte[] *buffer*, int *offset*, int *count* )

Reads PCM samples from the underlying buffer. If no data is available, returns silent samples.

##### Parameters

<i>buffer</i>	Buffer to which the data is copied to
<i>offset</i>	Offset for the data at the target buffer
<i>count</i>	Desired count of data

##### Returns



6.7.3.2 override void Hake\_WPF.AudioVideoManagers.BufferWaveStream.Write ( byte[] *buffer*, int *offset*, int *count* )

Writes the provided audio data to the outgoing PCM segment buffer.

## Parameters

<i>buffer</i>	Bytes to write
<i>offset</i>	Offset at which the bytes to write start
<i>count</i>	Count of the bytes to write<param>

## 6.7.4 Member Data Documentation

6.7.4.1 Queue<byte[]> Hake\_WPF.AudioVideoManagers.BufferWaveStream.completePcmSegments [private]

6.7.4.2 long Hake\_WPF.AudioVideoManagers.BufferWaveStream.defaultPosition = 0 [private]

6.7.4.3 long Hake\_WPF.AudioVideoManagers.BufferWaveStream.lengthInSamples = 1 [private]

6.7.4.4 byte [] Hake\_WPF.AudioVideoManagers.BufferWaveStream.overflowFromLastSegment [private]

6.7.4.5 object Hake\_WPF.AudioVideoManagers.BufferWaveStream.queueLock [private]

6.7.4.6 readonly WaveFormat Hake\_WPF.AudioVideoManagers.BufferWaveStream.waveFormat [private]

## 6.7.5 Property Documentation

6.7.5.1 override long Hake\_WPF.AudioVideoManagers.BufferWaveStream.Length [get]

6.7.5.2 override long Hake\_WPF.AudioVideoManagers.BufferWaveStream.Position [get], [set]

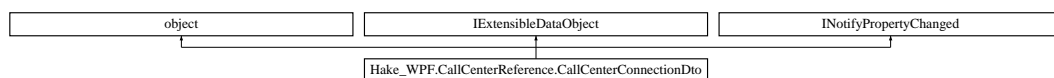
6.7.5.3 override WaveFormat Hake\_WPF.AudioVideoManagers.BufferWaveStream.WaveFormat [get]

The documentation for this class was generated from the following file:

- [BufferWaveStream.cs](#)

## 6.8 Hake\_WPF.CallCenterReference.CallCenterConnectionDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.CallCenterConnectionDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [ConnectionGuid](#) [get, set]
- string [UserName](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [ConnectionGuidField](#)
- string [UserNameField](#)

## 6.8.1 Member Function Documentation

6.8.1.1 void Hake\_WPF.CallCenterReference.CallCenterConnectionDto.RaisePropertyChanged ( string *propertyName* )  
[protected]

## 6.8.2 Member Data Documentation

6.8.2.1 string Hake\_WPF.CallCenterReference.CallCenterConnectionDto.ConnectionGuidField [private]

6.8.2.2 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.CallCenterConnectionDto.  
extensionDataField [private]

6.8.2.3 string Hake\_WPF.CallCenterReference.CallCenterConnectionDto.UserNameField [private]

## 6.8.3 Property Documentation

6.8.3.1 string Hake\_WPF.CallCenterReference.CallCenterConnectionDto.ConnectionGuid [get], [set]

6.8.3.2 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.CallCenterConnectionDto.  
ExtensionData [get], [set]

6.8.3.3 string Hake\_WPF.CallCenterReference.CallCenterConnectionDto.UserName [get], [set]

## 6.8.4 Event Documentation

6.8.4.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.CallCenterConnectionDto.  
PropertyChanged

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.9 Hake\_WPF.Conversion.CompressionHelper Class Reference

Class for static compression methods.

## Static Public Member Functions

- static byte[] [CompressGZip](#) (byte[] sourceData, bool useOptimalCompression)  
*Compresses the provided bytearray using the GZip algorithm provided by System.IO.Compression library.*
- static byte[] [DecompressGZip](#) (byte[] sourceData)  
*Decompresses the provided bytearray using the GZip algorithm provided by System.IO.Compression library.*

### 6.9.1 Detailed Description

Class for static compression methods.

<author>Veli-Mikko Puupponen</author> NOTICE: IO.Compression-library will not work, if the solution has Active Config set to any CPU. For phone, this should be ARM and for emulator, x86.

### 6.9.2 Member Function Documentation

6.9.2.1 static byte [] Hake\_WPF.Conversion.CompressionHelper.CompressGZip ( byte[] *sourceData*, bool *useOptimalCompression* ) [static]

Compresses the provided bytearray using the GZip algorithm provided by System.IO.Compression library.

Parameters

<i>sourceData</i>	Data to be compressed
<i>useOptimalCompression</i>	True, if using optimal compression method, otherwise using fastest

Returns

GZip compressed byte data

6.9.2.2 static byte [] Hake\_WPF.Conversion.CompressionHelper.DecompressGZip ( byte[] *sourceData* ) [static]

Decompresses the provided bytearray using the GZip algorithm provided by System.IO.Compression library.

Parameters

<i>sourceData</i>	Data to be decompressed
-------------------	-------------------------

Returns

GZip decompressed input byte data

The documentation for this class was generated from the following file:

- [CompressionHelper.cs](#)

## 6.10 Hake\_WPF.Connection Class Reference

Handles communication between server and this client. User must use connect method to connect server. After that user can use rest of the methods to manage connection. Also user should handle most of events, but atleast connectionupdatedevent.

### Public Member Functions

- delegate void [AssignmentUpdated](#) (ObservableCollection< [Assignment](#) > newConnections)  
*Invoked when connection is updated.*
- [Connection](#) (String username, String password)  
*Creates connection with credentials provided.*
- ObservableCollection< [Assignment](#) > [Connect](#) (String endpointAddress)  
*Creates receiver and connects to server with wcf and udp.*

- bool [UdpSendMedia](#) (MediaInformation mediaInfo, byte[] data, int originalLength)  
*Sends media to the server using the active UdpMediaClientSocket instance. Returns true, if the data is successfully queued for sending, otherwise returns false.*
- void [ChangePriority](#) ([Assignment](#) assignment, [Assignment.priorities](#) priority)  
*Changes priority to currentconnection and then updates that to connectionslist and finally to server. It makes the connection in new thread so it won't block ui thread.*
- ObservableCollection< [Assignment](#) > [ConvertConnectionsToAssignments](#) ([ConnectionDto](#)[] Connections)  
*Converts ConnectionDto to Assignment. If location is not set, uses null and do not assign accuracymeters or time from location. If device info is empty, uses empty DeviceInfoDto.*
- void [RequestAction](#) ([RemoteActionDto](#) action)  
*Requests action for current userconnection for currentconnection with given action. CurrentConnection needed or this does nothing.*
- void [RequestMeasurementData](#) ([MeasurementInstrumentDto](#) instrument)  
*If there is currentconnection requests measurements data start from server with given instrument.*
- void [CancelMeasurementData](#) ([MeasurementInstrumentDto](#) instrument)  
*Cancel request to server to stop measurement data sending. Current connection is needed for this.*
- bool [IsConnected](#) ()  
*Is Connection on.*
- ObservableCollection< [Assignment](#) > [GetAllConnections](#) ()  
*Should be used only one time to save locally and then use ActiveConnectionsUpdated to update those. Userconnection and converts server response to assignment list.*
- bool [Reconnect](#) ()  
*Reconnects to server.*
- bool [Disconnect](#) ()  
*Disconnects from server.*
- void [ProcessAssignment](#) (object assignment)  
*Changes assignments state to in progress to server. First it search right connection using guid and then makes the connection currentconnection.*
- void [SendTextMessage](#) (String message)  
*Makes new thread to send message to server so it won't block ui thread.*
- void [RequestAudioVideoMedia](#) ()  
*Makes new thread to request audio video so it won't block ui thread.*
- void [CancelAudioVideoMedia](#) ()  
*Makes new thread to request cancel of audio and video so it won't block ui thread.*
- async Task [PingAsync](#) (int interval)  
*Sends a keepalive message to server*

## Public Attributes

- [UdpMediaDataReceived](#) [UdpMediaDataReceivedEvent](#)

## Properties

- [AsyncReceiver](#) receiver [get, set]  
*Gets and sets AsyncReceiver.*

## Events

- [AssignmentUpdated](#) [AssignmentUpdatedEvent](#)

## Private Member Functions

- void [InitializeUdpClient](#) ()  
*Initializes a new `UdpMediaClientSocket` instance for media transfer from and to the server.*
- void [UdpMediaPacketReceivedHandler](#) (object sender, `MediaPacket` mediaPacket)  
*Handles incoming media packet events from the active `UdpMediaClientSocket` instance. Republishes them as `UdpMediaDataReceivedEvent`.*
- void [UdpSocketFailedEventHandler](#) (object sender)  
*Handles failure events of the active `UdpMediaClientSocket`. Currently only writes this information to the debug.*
- void [ChangePriorityThread](#) (object priority)  
*If there is no current connection this will not do anything. If there is it changes current connections priority and send it to server.*
- void [receiver\\_ConnectionsUpdatedEvent](#) (`ConnectionDto`[] newConnections)  
*Handles when connectionupdates are received. Updates connectionslist and invokes event for update with connectiondto's converted to assignments.*
- void [UpdateConnectionsList](#) (`ConnectionDto`[] newConnections)  
*Replaces connections with new connections by comparing their guid. If there is no connection with that guid, adds new connection.*
- void [SendMessage](#) (object message)  
*Send message to server.*
- void [RequestAudioVideoMediaThread](#) ()  
*Makes request to server for audio and video with hardcoded specs in `exampleMediaConfiguration`.*
- void [CancelAudioVideoMediaThread](#) ()  
*Request cancel to audio and video from server. It is done by `mediaConfigurationdto` and setting `enableaudio` and `enablepicture` to false.*

## Private Attributes

- const int `udpServerPort` = 15103
- const string `udpServerIp` = "130.234.9.165"
- `WcfCallCenterServiceClient` client
- string `guid`
- `InstanceContext` context
- `CallCenterConnectionDto` userConnection
- `ObservableCollection`  
    < `ConnectionDto` > `connectionDtos` = new `ObservableCollection`<`ConnectionDto`>()
- `ConnectionDto` currentConnection
- `UserCredentialsDto` userCredentialsDto
- `UdpMediaClientSocket` udpSocket
- bool `isConnected` = false

### 6.10.1 Detailed Description

Handles communication between server and this client. User must use connect method to connect server. After that user can use rest of the methods to manage connection. Also user should handle most of events, but atleast connectionupdatedevent.

<author>Atte Söderlund</author>

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 Hake\_WPF.Connection.Connection ( String username, String password )

Creates connection with credentials provided.

## Parameters

<i>username</i>	Username that connection uses
<i>password</i>	Password that connection uses

## 6.10.3 Member Function Documentation

6.10.3.1 `delegate void Hake_WPF.Connection.AssignmentUpdated ( ObservableCollection< Assignment > newConnections )`

Invoked when connection is updated.

## Parameters

<i>newConnections</i>	Connections as Assignments
-----------------------	----------------------------

6.10.3.2 `void Hake_WPF.Connection.CancelAudioVideoMedia ( )`

Makes new thread to request cancel of audio and video so it won't block ui thread.

6.10.3.3 `void Hake_WPF.Connection.CancelAudioVideoMediaThread ( ) [private]`

Request cancel to audio and video from server. It is done by mediaConfigurationdto and setting enableaudio and enablepicture to false.

6.10.3.4 `void Hake_WPF.Connection.CancelMeasurementData ( MeasurementInstrumentDto instrument )`

Cancel request to server to stop measurement data sending. Current connection is needed for this.

<author>Niko Mononen</author>

## Parameters

<i>instrument</i>	Instrument that is sending the data
-------------------	-------------------------------------

6.10.3.5 `void Hake_WPF.Connection.ChangePriority ( Assignment assignment, Assignment.priorities priority )`

Changes priority to currentconnection and then updates that to connectionslist and finally to server. It makes the connection in new thread so it won't block ui thread.

## Parameters

<i>assignment</i>	<a href="#">Assignment</a> that needs updating
<i>priority</i>	Priority that assignment is to be changed.

6.10.3.6 `void Hake_WPF.Connection.ChangePriorityThread ( object priority ) [private]`

If there is no current connection this will not do anything. If there is it changes current connections priority and send it to server.

## Parameters

<i>priority</i>	New priority
-----------------	--------------

#### 6.10.3.7 `ObservableCollection<Assignment> Hake_WPF.Connection.Connect ( String endpointAddress )`

Creates receiver and connects to server with wcf and udp.

##### Returns

ObservableCollection of connections as assignments.

#### 6.10.3.8 `ObservableCollection<Assignment> Hake_WPF.Connection.ConvertConnectionsToAssignments ( ConnectionDto[] Connections )`

Converts ConnectionDto to [Assignment](#). If location is not set, uses null and do not assign accuracymeters or time from location. If device info is empty, uses empty DeviceInfoDto.

##### Parameters

<a href="#">Connection</a>	ConnectionDto that needs converting to Assignment
----------------------------	---

##### Returns

Converted Assignment

#### 6.10.3.9 `bool Hake_WPF.Connection.Disconnect ( )`

Disconnects from server.

##### Returns

True when disconnect goes without error and false if there is any error.

#### 6.10.3.10 `ObservableCollection<Assignment> Hake_WPF.Connection.GetAllConnections ( )`

Should be used only one time to save locally and then use ActiveConnectionsUpdated to update those. Userconnection and converts server response to assignment list.

##### Returns

ObservableCollection of all connections as [Assignment](#)

#### 6.10.3.11 `void Hake_WPF.Connection.InitializeUdpClient ( ) [private]`

Initializes a new UdpMediaClientSocket instance for media transfer from and to the server.

#### 6.10.3.12 `bool Hake_WPF.Connection.IsConnected ( )`

Is [Connection](#) on.

##### Returns

Boolean of is connection established.



**6.10.3.13    async Task Hake\_WPF.Connection.PingAsync ( int *interval* )**

Sends a keepalive message to server

<author>Ilkka Rautiainen</author>

**Parameters**

<i>interval</i>	ping interval in seconds
-----------------	--------------------------

**Returns**

task

**6.10.3.14    void Hake\_WPF.Connection.ProcessAssignment ( object *assignment* )**

Changes assignments state to in progress to server. First it search right connection using guid and then makes the connection currentconnection.

**Parameters**

<i>assignment</i>	<a href="#">Assignment</a> that is to be set as in progress
-------------------	---

**6.10.3.15    void Hake\_WPF.Connection.receiver\_ConnectionsUpdatedEvent ( **ConnectionDto[]** *newConnections* )  
[private]**

Handles when connectionupdates are received. Updates connectionslist and invokes event for update with connectiondto's converted to assignments.

**Parameters**

<i>newConnections</i>	New connections
-----------------------	-----------------

**6.10.3.16    bool Hake\_WPF.Connection.Reconnect ( )**

Reconnects to server.

**Returns**

True when reconnect goes without error and false if there is any error.

**6.10.3.17    void Hake\_WPF.Connection.RequestAction ( **RemoteActionDto** *action* )**

Requests action for current userconnection for currentconnection with given action. CurrentConnection needed or this does nothing.

**Parameters**

<i>action</i>	What user wants action to be
---------------	------------------------------

**6.10.3.18    void Hake\_WPF.Connection.RequestAudioVideoMedia ( )**

Makes new thread to request audio video so it won't block ui thread.

6.10.3.19 void Hake\_WPF.Connection.RequestAudioVideoMediaThread ( ) [private]

Makes request to server for audio and video with hardcoded specs in exampleMediaConfiguration.

6.10.3.20 void Hake\_WPF.Connection.RequestMeasurementData ( MeasurementInstrumentDto *instrument* )

If there is currentconnection requests measurements data start from server with given instrument.

<author>Niko Mononen</author>

#### Parameters

<i>instrument</i>	Instrument that provides data
-------------------	-------------------------------

6.10.3.21 void Hake\_WPF.Connection.SendMessage ( object *message* ) [private]

Send message to server.

#### Parameters

<i>message</i>	Message to be send
----------------	--------------------

6.10.3.22 void Hake\_WPF.Connection.SendTextMessage ( String *message* )

Makes new thread to send message to server so it won't block ui thread.

#### Parameters

<i>message</i>	Message to be send
----------------	--------------------

6.10.3.23 void Hake\_WPF.Connection.UdpMediaPacketReceivedHandler ( object *sender*, MediaPlayer *mediaPacket* )  
[private]

Handles incoming media packet events from the active UdpMediaClientSocket instance. Republishes them as UdpMediaDataReceivedEvent.

#### Parameters

<i>sender</i>	UdpMediaDataReceivedEvent event punlishing this event
<i>mediaPacket</i>	The received MediaPlayer instance

6.10.3.24 bool Hake\_WPF.Connection.UdpSendMedia ( MediaInformation *mediaInfo*, byte[] *data*, int *originalLength* )

Sends media to the server using the active UdpMediaClientSocket instance. Returns true, is the data is successfully queued for sending, otherwise returns false.

#### Parameters

<i>mediaInfo</i>	Description of the media
<i>data</i>	Bytes constituting the media data
<i>originalLength</i>	Length of the media prior to encoding for such encodings that require the length for decompression.

#### Returns

True, if data queued for sending, otherwise false

6.10.3.25 void Hake\_WPF.Connection.UdpSocketFailedEventHandler ( object *sender* ) [private]

Handles failure events of the active UdpMediaClientSocket. Currently only writes this information to the debug.

Parameters

<i>sender</i>	UdpMediaClientSocket instance sending the event
---------------	---

6.10.3.26 void Hake\_WPF.Connection.UpdateConnectionsList ( ConnectionDto[] *newConnections* ) [private]

Replaces connections with new connections by comparing their guid. If there is no connection with that guid, adds new connection.

Parameters

<i>newConnections</i>	New connetions
-----------------------	----------------

## 6.10.4 Member Data Documentation

6.10.4.1 WcfCallCenterServiceClient Hake\_WPF.Connection.client [private]

6.10.4.2 ObservableCollection<ConnectionDto> Hake\_WPF.Connection.connectionDtos = new ObservableCollection<ConnectionDto>() [private]

6.10.4.3 InstanceContext Hake\_WPF.Connection.context [private]

6.10.4.4 ConnectionDto Hake\_WPF.Connection.currentConnection [private]

6.10.4.5 string Hake\_WPF.Connection.guid [private]

6.10.4.6 bool Hake\_WPF.Connection.isConnected = false [private]

6.10.4.7 UdpMediaDataReceived Hake\_WPF.Connection.UdpMediaDataReceivedEvent

6.10.4.8 const string Hake\_WPF.Connection.udpServerIp = "130.234.9.165" [private]

6.10.4.9 const int Hake\_WPF.Connection.udpServerPort = 15103 [private]

6.10.4.10 UdpMediaClientSocket Hake\_WPF.Connection.udpSocket [private]

6.10.4.11 CallCenterConnectionDto Hake\_WPF.Connection.userConnection [private]

6.10.4.12 UserCredentialsDto Hake\_WPF.Connection.userCredentialsDto [private]

## 6.10.5 Property Documentation

6.10.5.1 AsyncReceiver Hake\_WPF.Connection.receiver [get],[set]

Gets and sets [AsyncReceiver](#).

## 6.10.6 Event Documentation

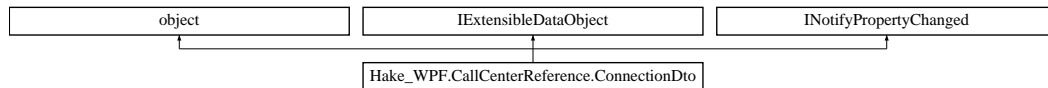
6.10.6.1 AssignmentUpdated Hake\_WPF.Connection.AssignmentUpdatedEvent

The documentation for this class was generated from the following file:

- [Connection.cs](#)

## 6.11 Hake\_WPF.CallCenterReference.ConnectionDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.ConnectionDto:



### Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

### Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- System.DateTime [ArrivalTime](#) [get, set]
- [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto\[\]](#) [AttachedCallCenterConnections](#) [get, set]
- string [ConnectionGuid](#) [get, set]
- [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto](#) [ConnectionPriority](#) [get, set]
- [Hake\\_WPF.CallCenterReference.ConnectionStateDto](#) [ConnectionState](#) [get, set]
- [Hake\\_WPF.CallCenterReference.EmergencyTypeDto](#) [EmergencyType](#) [get, set]
- [Hake\\_WPF.CallCenterReference.LocationInformationDto\[\]](#) [LocationInformation](#) [get, set]
- [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto\[\]](#) [MeasurementInstruments](#) [get, set]
- [Hake\\_WPF.CallCenterReference.MobileDeviceInformationDto\[\]](#) [MobileDeviceInformation](#) [get, set]
- [Hake\\_WPF.CallCenterReference.PersonalInformationDto](#) [PersonalInformation](#) [get, set]
- bool [RequestedNoSound](#) [get, set]
- [Hake\\_WPF.CallCenterReference.TextMessageDto\[\]](#) [TextMessages](#) [get, set]
- bool [clientDisconnected](#) [get, set]

### Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

### Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- System.DateTime [ArrivalTimeField](#)
- [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto\[\]](#) [AttachedCallCenterConnectionsField](#)
- string [ConnectionGuidField](#)
- [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto](#) [ConnectionPriorityField](#)
- [Hake\\_WPF.CallCenterReference.ConnectionStateDto](#) [ConnectionStateField](#)
- [Hake\\_WPF.CallCenterReference.EmergencyTypeDto](#) [EmergencyTypeField](#)
- [Hake\\_WPF.CallCenterReference.LocationInformationDto\[\]](#) [LocationInformationField](#)
- [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto\[\]](#) [MeasurementInstrumentsField](#)
- [Hake\\_WPF.CallCenterReference.MobileDeviceInformationDto\[\]](#) [MobileDeviceInformationField](#)
- [Hake\\_WPF.CallCenterReference.PersonalInformationDto](#) [PersonalInformationField](#)
- bool [RequestedNoSoundField](#)
- [Hake\\_WPF.CallCenterReference.TextMessageDto\[\]](#) [TextMessagesField](#)
- bool [clientDisconnectedField](#)

### 6.11.1 Member Function Documentation

6.11.1.1 void Hake\_WPF.CallCenterReference.ConnectionDto.RaisePropertyChanged ( string *propertyName* )  
[protected]

### 6.11.2 Member Data Documentation

6.11.2.1 System.DateTime Hake\_WPF.CallCenterReference.ConnectionDto.ArrivalTimeField [private]

6.11.2.2 Hake\_WPF.CallCenterReference.CallCenterConnectionDto [] Hake\_WPF.CallCenterReference.ConnectionDto.AttachedCallCenterConnectionsField [private]

6.11.2.3 bool Hake\_WPF.CallCenterReference.ConnectionDto.clientDisconnectedField [private]

6.11.2.4 string Hake\_WPF.CallCenterReference.ConnectionDto.ConnectionGuidField [private]

6.11.2.5 Hake\_WPF.CallCenterReference.ConnectionPriorityDto Hake\_WPF.CallCenterReference.ConnectionDto.ConnectionPriorityField [private]

6.11.2.6 Hake\_WPF.CallCenterReference.ConnectionStateDto Hake\_WPF.CallCenterReference.ConnectionDto.ConnectionStateField [private]

6.11.2.7 Hake\_WPF.CallCenterReference.EmergencyTypeDto Hake\_WPF.CallCenterReference.ConnectionDto.EmergencyTypeField [private]

6.11.2.8 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.ConnectionDto.extensionDataField [private]

6.11.2.9 Hake\_WPF.CallCenterReference.LocationInformationDto [] Hake\_WPF.CallCenterReference.ConnectionDto.LocationInformationField [private]

6.11.2.10 Hake\_WPF.CallCenterReference.MeasurementInstrumentDto [] Hake\_WPF.CallCenterReference.ConnectionDto.MeasurementInstrumentsField [private]

6.11.2.11 Hake\_WPF.CallCenterReference.MobileDeviceInformationDto [] Hake\_WPF.CallCenterReference.ConnectionDto.MobileDeviceInformationField [private]

6.11.2.12 Hake\_WPF.CallCenterReference.PersonalInformationDto Hake\_WPF.CallCenterReference.ConnectionDto.PersonalInformationField [private]

6.11.2.13 bool Hake\_WPF.CallCenterReference.ConnectionDto.RequestedNoSoundField [private]

6.11.2.14 Hake\_WPF.CallCenterReference.TextMessageDto [] Hake\_WPF.CallCenterReference.ConnectionDto.TextMessagesField [private]

### 6.11.3 Property Documentation

6.11.3.1 System.DateTime Hake\_WPF.CallCenterReference.ConnectionDto.ArrivalTime [get], [set]

6.11.3.2 Hake\_WPF.CallCenterReference.CallCenterConnectionDto [] Hake\_WPF.CallCenterReference.ConnectionDto.AttachedCallCenterConnections [get], [set]

6.11.3.3 bool Hake\_WPF.CallCenterReference.ConnectionDto.clientDisconnected [get], [set]

6.11.3.4 string Hake\_WPF.CallCenterReference.ConnectionDto.ConnectionGuid [get], [set]

- 6.11.3.5 **Hake\_WPF.CallCenterReference.ConnectionPriorityDto** **Hake\_WPF.CallCenterReference.ConnectionDto**.↔  
**ConnectionPriority** [get], [set]
- 6.11.3.6 **Hake\_WPF.CallCenterReference.ConnectionStateDto** **Hake\_WPF.CallCenterReference.ConnectionDto**.↔  
**ConnectionState** [get], [set]
- 6.11.3.7 **Hake\_WPF.CallCenterReference.EmergencyTypeDto** **Hake\_WPF.CallCenterReference.ConnectionDto**.↔  
**EmergencyType** [get], [set]
- 6.11.3.8 **System.Runtime.Serialization.ExtensionDataObject** **Hake\_WPF.CallCenterReference.ConnectionDto.ExtensionData**  
[get], [set]
- 6.11.3.9 **Hake\_WPF.CallCenterReference.LocationInformationDto** [] **Hake\_WPF.CallCenterReference.Connection**↔  
**Dto.LocationInformation** [get], [set]
- 6.11.3.10 **Hake\_WPF.CallCenterReference.MeasurementInstrumentDto** [] **Hake\_WPF.CallCenterReference**.↔  
**ConnectionDto.MeasurementInstruments** [get], [set]
- 6.11.3.11 **Hake\_WPF.CallCenterReference.MobileDeviceInformationDto** [] **Hake\_WPF.CallCenterReference**.↔  
**ConnectionDto.MobileDeviceInformation** [get], [set]
- 6.11.3.12 **Hake\_WPF.CallCenterReference.PersonalInformationDto** **Hake\_WPF.CallCenterReference.Connection**↔  
**Dto.PersonalInformation** [get], [set]
- 6.11.3.13 **bool** **Hake\_WPF.CallCenterReference.ConnectionDto.RequestedNoSound** [get], [set]
- 6.11.3.14 **Hake\_WPF.CallCenterReference.TextMessageDto** [] **Hake\_WPF.CallCenterReference.ConnectionDto.Text**↔  
**Messages** [get], [set]

## 6.11.4 Event Documentation

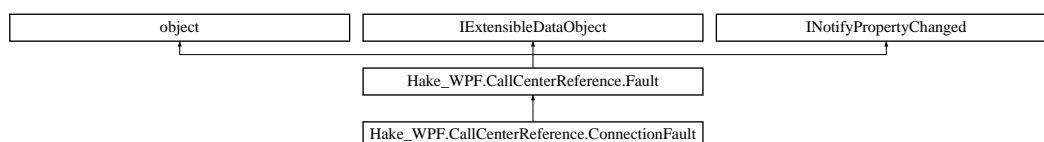
- 6.11.4.1 **System.ComponentModel.PropertyChangedEventHandler** **Hake\_WPF.CallCenterReference.ConnectionDto.Property**↔  
**Changed**

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.12 Hake\_WPF.CallCenterReference.ConnectionFault Class Reference

Inheritance diagram for **Hake\_WPF.CallCenterReference.ConnectionFault**:



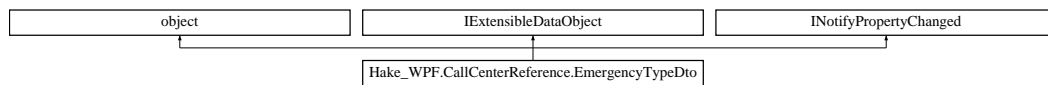
### Additional Inherited Members

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.13 Hake\_WPF.CallCenterReference.EmergencyTypeDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.EmergencyTypeDto:



### Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

### Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [TypeName](#) [get, set]

### Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

### Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [TypeNameField](#)

#### 6.13.1 Member Function Documentation

- 6.13.1.1 void `Hake_WPF.CallCenterReference.EmergencyTypeDto.RaisePropertyChanged ( string propertyName )`  
[protected]

#### 6.13.2 Member Data Documentation

- 6.13.2.1 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.EmergencyTypeDto.extensionDataField` [private]

- 6.13.2.2 string `Hake_WPF.CallCenterReference.EmergencyTypeDto.TypeNameField` [private]

#### 6.13.3 Property Documentation

- 6.13.3.1 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.EmergencyTypeDto.ExtensionData` [get], [set]

- 6.13.3.2 string `Hake_WPF.CallCenterReference.EmergencyTypeDto.TypeName` [get], [set]

#### 6.13.4 Event Documentation

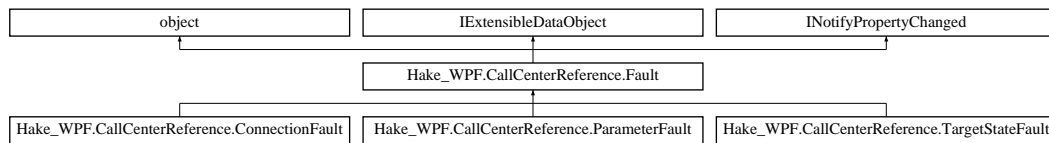
- 6.13.4.1 System.ComponentModel.PropertyChangedEventHandler `Hake_WPF.CallCenterReference.EmergencyTypeDto.PropertyChanged`

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.14 Hake\_WPF.CallCenterReference.Fault Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.Fault:



### Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

### Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [Cause](#) [get, set]
- string [Detail](#) [get, set]

### Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

### Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [CauseField](#)
- string [DetailField](#)

### 6.14.1 Member Function Documentation

- 6.14.1.1 void `Hake_WPF.CallCenterReference.Fault.RaisePropertyChanged ( string propertyName )` [protected]

### 6.14.2 Member Data Documentation

- 6.14.2.1 string `Hake_WPF.CallCenterReference.Fault.CauseField` [private]
- 6.14.2.2 string `Hake_WPF.CallCenterReference.Fault.DetailField` [private]
- 6.14.2.3 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.Fault.extensionDataField` [private]

### 6.14.3 Property Documentation

- 6.14.3.1 string `Hake_WPF.CallCenterReference.Fault.Cause` [get], [set]
- 6.14.3.2 string `Hake_WPF.CallCenterReference.Fault.Detail` [get], [set]
- 6.14.3.3 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.Fault.ExtensionData` [get], [set]



## 6.14.4 Event Documentation

### 6.14.4.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.Fault.PropertyChanged

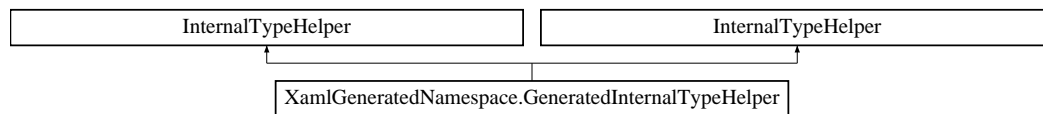
The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.15 XamlGeneratedNamespace.GeneratedInternalTypeHelper Class Reference

### [GeneratedInternalTypeHelper](#)

Inheritance diagram for XamlGeneratedNamespace.GeneratedInternalTypeHelper:



### Protected Member Functions

- override object [CreateInstance](#) (System.Type type, System.Globalization.CultureInfo culture)  
*CreateInstance*
- override object [GetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)  
*GetPropertyValue*
- override void [SetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)  
*SetPropertyValue*
- override System.Delegate [CreateDelegate](#) (System.Type delegateType, object target, string handler)  
*CreateDelegate*
- override void [AddEventHandler](#) (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)  
*AddEventHandler*
- override object [CreateInstance](#) (System.Type type, System.Globalization.CultureInfo culture)  
*CreateInstance*
- override object [GetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, System.Globalization.CultureInfo culture)  
*GetPropertyValue*
- override void [SetPropertyValue](#) (System.Reflection.PropertyInfo propertyInfo, object target, object value, System.Globalization.CultureInfo culture)  
*SetPropertyValue*
- override System.Delegate [CreateDelegate](#) (System.Type delegateType, object target, string handler)  
*CreateDelegate*
- override void [AddEventHandler](#) (System.Reflection.EventInfo eventInfo, object target, System.Delegate handler)  
*AddEventHandler*

### 6.15.1 Detailed Description

### [GeneratedInternalTypeHelper](#)

## 6.15.2 Member Function Documentation

6.15.2.1 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler ( System.Reflection.EventInfo *eventInfo*, object *target*, System.Delegate *handler* ) [protected]

AddEventHandler

6.15.2.2 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.AddEventHandler ( System.Reflection.EventInfo *eventInfo*, object *target*, System.Delegate *handler* ) [protected]

AddEventHandler

6.15.2.3 override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate ( System.Type *delegateType*, object *target*, string *handler* ) [protected]

CreateDelegate

6.15.2.4 override System.Delegate XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateDelegate ( System.Type *delegateType*, object *target*, string *handler* ) [protected]

CreateDelegate

6.15.2.5 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance ( System.Type *type*, System.Globalization.CultureInfo *culture* ) [protected]

CreateInstance

6.15.2.6 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.CreateInstance ( System.Type *type*, System.Globalization.CultureInfo *culture* ) [protected]

CreateInstance

6.15.2.7 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.GetPropertyValue ( System.Reflection.PropertyInfo *propertyInfo*, object *target*, System.Globalization.CultureInfo *culture* ) [protected]

GetPropertyValue

6.15.2.8 override object XamlGeneratedNamespace.GeneratedInternalTypeHelper.GetPropertyValue ( System.Reflection.PropertyInfo *propertyInfo*, object *target*, System.Globalization.CultureInfo *culture* ) [protected]

GetPropertyValue

6.15.2.9 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue ( System.Reflection.PropertyInfo *propertyInfo*, object *target*, object *value*, System.Globalization.CultureInfo *culture* ) [protected]

SetPropertyValue

6.15.2.10 override void XamlGeneratedNamespace.GeneratedInternalTypeHelper.SetPropertyValue (   
 System.Reflection.PropertyInfo *propertyInfo*, object *target*, object *value*, System.Globalization.CultureInfo *culture* )   
 [protected]

SetPropertyValue

The documentation for this class was generated from the following files:

- [GeneratedInternalTypeHelper.g.cs](#)
- [GeneratedInternalTypeHelper.g.i.cs](#)

## 6.16 GraphClass.Graph Class Reference

Adds graph to given grid. user can add points, set bytespersecond and scroll to end.

### Public Member Functions

- [Graph](#) (Grid grid)  
*Makes graphcanvas to given grid with black background and initializes green line that connects points that are added by addpoint method.*
- void [setBytesPerSecond](#) (int bps)  
*Sets BytesPerSecond.*
- void [AddPoint](#) (int y)  
*Adds point to graph.*
- void [ScrollToEnd](#) ()  
*Scrolls to end of the graph.*

### Private Attributes

- const double [ScaleRate](#) = 1.1

### Static Private Attributes

- static Canvas [GraphCanvas](#)
- static ScrollViewer [GraphContainer](#)
- static ScaleTransform [GraphScaleTransform](#)
- static Polyline [GraphLine](#)
- static int [GraphLineX](#) = 0
- static Color [GraphColor](#) = Color.FromArgb(200, 255, 187, 187)
- static Color [GraphSecondColor](#) = Color.FromArgb(150, 255, 229, 229)
- static SolidColorBrush [color1](#) = new SolidColorBrush([GraphColor](#))
- static SolidColorBrush [color2](#) = new SolidColorBrush([GraphSecondColor](#))
- static int [scale](#) = 1000
- static int [BytesPerSecond](#) = 50

### 6.16.1 Detailed Description

Adds graph to given grid. user can add points, set bytespersecond and scroll to end.

<author>Niko Mononen</author>

## 6.16.2 Constructor & Destructor Documentation

### 6.16.2.1 GraphClass.Graph.Graph ( Grid *grid* )

Makes graphcanvas to given grid with black background and initializes green line that connects points that are added by addpoint method.

## Parameters

<i>grid</i>	Grid where you want this graph
-------------	--------------------------------

## 6.16.3 Member Function Documentation

6.16.3.1 void GraphClass.Graph.AddPoint ( int *y* )

Adds point to graph.

## Parameters

<i>y</i>	y-point of the graph
----------	----------------------

## 6.16.3.2 void GraphClass.Graph.ScrollToEnd ( )

Scrolls to end of the graph.

6.16.3.3 void GraphClass.Graph.setBytesPerSecond ( int *bps* )

Sets BytesPerSecond.

## Parameters

<i>bps</i>	Bytes per second
------------	------------------

## 6.16.4 Member Data Documentation

6.16.4.1 int GraphClass.Graph.BytesPerSecond = 50 [static], [private]

6.16.4.2 SolidColorBrush GraphClass.Graph.color1 = new SolidColorBrush(GraphColor) [static], [private]

6.16.4.3 SolidColorBrush GraphClass.Graph.color2 = new SolidColorBrush(GraphSecondColor) [static], [private]

6.16.4.4 Canvas GraphClass.Graph.GraphCanvas [static], [private]

6.16.4.5 Color GraphClass.Graph.GraphColor = Color.FromArgb(200, 255, 187, 187) [static], [private]

6.16.4.6 ScrollViewer GraphClass.Graph.GraphContainer [static], [private]

6.16.4.7 Polyline GraphClass.Graph.GraphLine [static], [private]

6.16.4.8 int GraphClass.Graph.GraphLineX = 0 [static], [private]

6.16.4.9 ScaleTransform GraphClass.Graph.GraphScaleTransform [static], [private]

6.16.4.10 Color GraphClass.Graph.GraphSecondColor = Color.FromArgb(150, 255, 229, 229) [static], [private]

6.16.4.11 int GraphClass.Graph.scale = 1000 [static], [private]

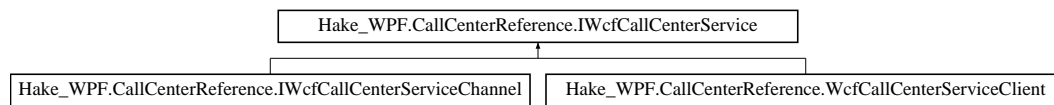
6.16.4.12 const double GraphClass.Graph.ScaleRate = 1.1 [private]

The documentation for this class was generated from the following file:

- [GraphClass.cs](#)

## 6.17 Hake\_WPF.CallCenterReference.IWcfCallCenterService Interface Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.IWcfCallCenterService:



### Public Member Functions

- [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) [Connect](#) ([Hake\\_WPF.CallCenterReference.IWcfCallCenterServiceChannel](#) credentials)
- [System.Threading.Tasks.Task](#) [ConnectAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) credentials)
- [void](#) [Reconnect](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user)
- [System.Threading.Tasks.Task](#) [ReconnectAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user)
- [void](#) [Disconnect](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user)
- [System.Threading.Tasks.Task](#) [DisconnectAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user)
- [Hake\\_WPF.CallCenterReference.ConnectionDto\[\]](#) [GetActiveConnections](#) ([Hake\\_WPF.CallCenterReference.IWcfCallCenterServiceChannel](#) user)
- [System.Threading.Tasks.Task](#) [GetActiveConnectionsAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user)
- [void](#) [OpenConnectionForProcessing](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [System.Threading.Tasks.Task](#) [OpenConnectionForProcessingAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [void](#) [TransferConnection](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto\[\]](#) targetCallCenterConnections)
- [System.Threading.Tasks.Task](#) [TransferConnectionAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto\[\]](#) targetCallCenterConnections)
- [void](#) [SetConnectionPriority](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [System.Threading.Tasks.Task](#) [SetConnectionPriorityAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [void](#) [MoveConnectionToHold](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [System.Threading.Tasks.Task](#) [MoveConnectionToHoldAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [void](#) [MarkProcessedCloseConnection](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [System.Threading.Tasks.Task](#) [MarkProcessedCloseConnectionAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- [void](#) [RequestRemoteAction](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.RemoteActionDto](#) action)
- [System.Threading.Tasks.Task](#) [RequestRemoteActionAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.RemoteActionDto](#) action)
- [void](#) [RequestMediaUpstreaming](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MediaConfigurationDto](#) mediaConfiguration)

- System.Threading.Tasks.Task [RequestMediaUpstreamingAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MediaConfigurationDto](#) mediaConfiguration)
- void [RequestMediaDownstreaming](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, string mediaUrl)
- System.Threading.Tasks.Task [RequestMediaDownstreamingAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, string mediaUrl)
- void [RequestStartMeasurement](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- System.Threading.Tasks.Task [RequestStartMeasurementAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- void [RequestStopMeasurement](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- System.Threading.Tasks.Task [RequestStopMeasurementAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- void [SendTextMessage](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.TextMessageDto](#) textMessage)
- System.Threading.Tasks.Task [SendTextMessageAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.TextMessageDto](#) textMessage)
- bool [UploadMediaSegment](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.MediaInformationDto](#) mediaInfo, byte[] mediaData)
- System.Threading.Tasks.Task< bool > [UploadMediaSegmentAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.MediaInformationDto](#) mediaInfo, byte[] mediaData)
- int [Ping](#) (int pingSequence)
- System.Threading.Tasks.Task< int > [PingAsync](#) (int pingSequence)

## 6.17.1 Member Function Documentation

6.17.1.1 [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) [Hake\\_WPF.CallCenterReference.IWcfCallCenterService.Connect](#) ([Hake\\_WPF.CallCenterReference.UserCredentialsDto](#) *credentials*)

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.2 System.Threading.Tasks.Task<[Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#)> [Hake\\_WPF.CallCenterReference.IWcfCallCenterService.ConnectAsync](#) ([Hake\\_WPF.CallCenterReference.UserCredentialsDto](#) *credentials*)

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.3 void [Hake\\_WPF.CallCenterReference.IWcfCallCenterService.Disconnect](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) *user*)

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.4 **System.Threading.Tasks.Task** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.DisconnectAsync (**  
**Hake\_WPF.CallCenterReference.CallCenterConnectionDto user )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.5 **Hake\_WPF.CallCenterReference.ConnectionDto []** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.GetActiveConnections (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user**  
**)**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.6 **System.Threading.Tasks.Task<****Hake\_WPF.CallCenterReference.ConnectionDto[]>**  
**Hake\_WPF.CallCenterReference.IWcfCallCenterService.GetActiveConnectionsAsync (**  
**Hake\_WPF.CallCenterReference.CallCenterConnectionDto user )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.7 **void** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.MarkProcessedCloseConnection (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.8 **System.Threading.Tasks.Task** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.MarkProcessedCloseConnectionAsync (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user,**  
**Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.9 **void** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.MoveConnectionToHold (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.10 **System.Threading.Tasks.Task** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.MoveConnectionToHoldAsync (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user,**  
**Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.11 **void** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.OpenConnectionForProcessing (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.12 **System.Threading.Tasks.Task** **Hake\_WPF.CallCenterReference.IWcfCallCenterService.OpenConnectionForProcessingAsync (** **Hake\_WPF.CallCenterReference.CallCenterConnectionDto user,**  
**Hake\_WPF.CallCenterReference.ConnectionDto connection )**

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).



6.17.1.13 `int Hake_WPF.CallCenterReference.IWcfCallCenterService.Ping ( int pingSequence )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.14 `System.Threading.Tasks.Task<int> Hake_WPF.CallCenterReference.IWcfCallCenterService.PingAsync ( int pingSequence )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.15 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.Reconnect ( Hake_WPF.CallCenterReference.IWcfCallCenterReference.CallCenterConnectionDto user )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.16 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.ReconnectAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.17 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestMediaDownstreaming ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, string mediaUrl )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.18 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestMediaDownstreamingAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, string mediaUrl )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.19 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestMediaUpstreaming ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MediaConfigurationDto mediaConfiguration )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.20 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestMediaUpstreamingAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MediaConfigurationDto mediaConfiguration )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.21 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestRemoteAction ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.RemoteActionDto action )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.22 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestRemoteActionAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.RemoteActionDto action )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.23 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestStartMeasurement ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.24 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestStartMeasurementAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.25 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestStopMeasurement ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.26 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.RequestStopMeasurementAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.27 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.SendTextMessage ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.TextMessageDto textMessage )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.28 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.SendTextMessageAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection, Hake_WPF.CallCenterReference.TextMessageDto textMessage )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.29 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.SetConnectionPriority ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.30 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.SetConnection←  
PriorityAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.31 `void Hake_WPF.CallCenterReference.IWcfCallCenterService.TransferConnection (  
Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WP←  
F.CallCenterReference.CallCenterConnectionDto[] targetCallCenterConnections  
)`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.32 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.IWcfCallCenterService.Transfer←  
ConnectionAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.CallCenterConnectionDto[] targetCallCenterConnections )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.33 `bool Hake_WPF.CallCenterReference.IWcfCallCenterService.UploadMediaSegment ( Hake_WPF.CallCenter←  
Reference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.MediaInformationDto  
mediaInfo, byte[] mediaData )`

Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

6.17.1.34 `System.Threading.Tasks.Task<bool> Hake_WPF.CallCenterReference.IWcfCallCenterService.Upload←  
MediaSegmentAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.MediaInformationDto mediaInfo, byte[] mediaData )`

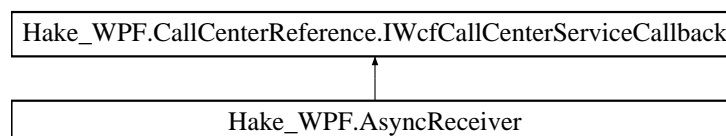
Implemented in [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#).

The documentation for this interface was generated from the following file:

- [Reference.cs](#)

## 6.18 Hake\_WPF.CallCenterReference.IWcfCallCenterServiceCallback Interface Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.IWcfCallCenterServiceCallback:



### Public Member Functions

- void [ActiveConnectionsUpdated](#) ([Hake\\_WPF.CallCenterReference.ConnectionDto\[\]](#) updatedConnections)
- void [AudioVideoReceived](#) (string sourceGuid, [Hake\\_WPF.CallCenterReference.MediaInformationDto](#) mediaInfo, byte[] mediaData)
- void [MeasurementDataReceived](#) (string sourceGuid, [Hake\\_WPF.CallCenterReference.Measurement←  
InstrumentDto](#) instrument, byte[] measurementData)

### 6.18.1 Member Function Documentation

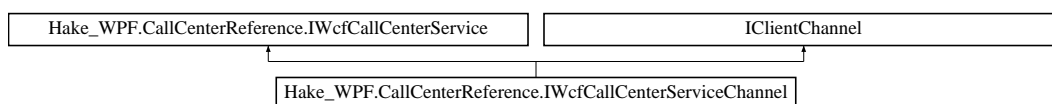
- 6.18.1.1 void Hake\_WPF.CallCenterReference.IWcfCallCenterServiceCallback.ActiveConnectionsUpdated ( Hake\_WPF.CallCenterReference.ConnectionDto[] *updatedConnections* )
- 6.18.1.2 void Hake\_WPF.CallCenterReference.IWcfCallCenterServiceCallback.AudioVideoReceived ( string *sourceGuid*, Hake\_WPF.CallCenterReference.MediaInformationDto *medialInfo*, byte[] *mediaData* )
- 6.18.1.3 void Hake\_WPF.CallCenterReference.IWcfCallCenterServiceCallback.MeasurementDataReceived ( string *sourceGuid*, Hake\_WPF.CallCenterReference.MeasurementInstrumentDto *instrument*, byte[] *measurementData* )

The documentation for this interface was generated from the following file:

- [Reference.cs](#)

## 6.19 Hake\_WPF.CallCenterReference.IWcfCallCenterServiceChannel Interface Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.IWcfCallCenterServiceChannel:



### Additional Inherited Members

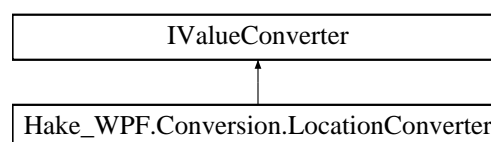
The documentation for this interface was generated from the following file:

- [Reference.cs](#)

## 6.20 Hake\_WPF.Conversion.LocationConverter Class Reference

Converts location to string

Inheritance diagram for Hake\_WPF.Conversion.LocationConverter:



### Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)  
*Casts value to LocationInformation and then turn it to string where is 4 decimals.*
- object [ConvertBack](#) (object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)  
*Do not use. Always returns null.*

### 6.20.1 Detailed Description

Converts location to string

<author>Atte Söderlund</author>

### 6.20.2 Member Function Documentation

6.20.2.1 object Hake\_WPF.Conversion.LocationConverter.Convert ( object *value*, Type *targetType*, object *parameter*, System.Globalization.CultureInfo *culture* )

Casts value to LocationInformation and then turn it to string where is 4 decimals.

Parameters

<i>value</i>	LocationInformationDto that need to be converted to string
<i>targetType</i>	Is not in use
<i>parameter</i>	Is not in use
<i>culture</i>	Is not in use

Returns

Location as string with 4 decimals on longitude and latitude

6.20.2.2 object Hake\_WPF.Conversion.LocationConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, System.Globalization.CultureInfo *culture* )

Do not use. Always returns null.

Returns

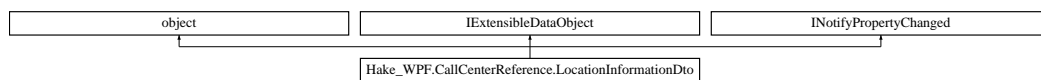
null

The documentation for this class was generated from the following file:

- [LocationConverter.cs](#)

## 6.21 Hake\_WPF.CallCenterReference.LocationInformationDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.LocationInformationDto:



### Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

### Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- double [AccuracyMeters](#) [get, set]

- System.DateTimeOffset [AcquisitionTime](#) [get, set]
- double [Latitude](#) [get, set]
- [Hake\\_WPF.CallCenterReference.LocationTypeDto LocationType](#) [get, set]
- double [Longitude](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- double [AccuracyMetersField](#)
- System.DateTimeOffset [AcquisitionTimeField](#)
- double [LatitudeField](#)
- [Hake\\_WPF.CallCenterReference.LocationTypeDto LocationTypeField](#)
- double [LongitudeField](#)

### 6.21.1 Member Function Documentation

- 6.21.1.1 void [Hake\\_WPF.CallCenterReference.LocationInformationDto.RaisePropertyChanged \( string propertyName \)](#)  
[protected]

### 6.21.2 Member Data Documentation

- 6.21.2.1 double [Hake\\_WPF.CallCenterReference.LocationInformationDto.AccuracyMetersField](#) [private]
- 6.21.2.2 System.DateTimeOffset [Hake\\_WPF.CallCenterReference.LocationInformationDto.AcquisitionTimeField](#)  
[private]
- 6.21.2.3 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.LocationInformationDto.↔  
extensionDataField](#) [private]
- 6.21.2.4 double [Hake\\_WPF.CallCenterReference.LocationInformationDto.LatitudeField](#) [private]
- 6.21.2.5 [Hake\\_WPF.CallCenterReference.LocationTypeDto Hake\\_WPF.CallCenterReference.LocationInformation↔  
Dto.LocationTypeField](#) [private]
- 6.21.2.6 double [Hake\\_WPF.CallCenterReference.LocationInformationDto.LongitudeField](#) [private]

### 6.21.3 Property Documentation

- 6.21.3.1 double [Hake\\_WPF.CallCenterReference.LocationInformationDto.AccuracyMeters](#) [get], [set]
- 6.21.3.2 System.DateTimeOffset [Hake\\_WPF.CallCenterReference.LocationInformationDto.AcquisitionTime](#) [get], [set]
- 6.21.3.3 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.LocationInformationDto.↔  
ExtensionData](#) [get], [set]
- 6.21.3.4 double [Hake\\_WPF.CallCenterReference.LocationInformationDto.Latitude](#) [get], [set]
- 6.21.3.5 [Hake\\_WPF.CallCenterReference.LocationTypeDto Hake\\_WPF.CallCenterReference.LocationInformation↔  
Dto.LocationType](#) [get], [set]

6.21.3.6 double Hake\_WPF.CallCenterReference.LocationInformationDto.Longitude [get], [set]

## 6.21.4 Event Documentation

6.21.4.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.LocationInformationDto.  
PropertyChanged

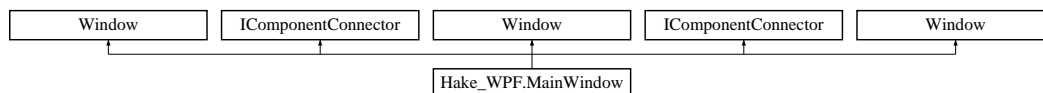
The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.22 Hake\_WPF.MainWindow Class Reference

Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.

Inheritance diagram for Hake\_WPF.MainWindow:



### Public Member Functions

- [MainWindow](#) ()  
*Initializes components and focuses map. Also connects to server by endpointAddress that is in settings or by default. Then adds eventhandlers and informs user if conenction is not established in case there is some problem. Also starts Pinger that pings server.*
- void [InitializeComponent](#) ()  
*InitializeComponent*
- void [InitializeComponent](#) ()  
*InitializeComponent*

### Protected Member Functions

- override void [OnClosing](#) (System.ComponentModel.CancelEventArgs e)  
*Disconnects before closing program. Also if mapwindows or/and settingswindow is open, closes those too.*

### Private Member Functions

- void [mapController\\_MapWindowClosingEvent](#) ()  
*Enables MapTopOwnWindowButton when [MapWindow](#) is closing.*
- void [MapToOwnWindow\\_Click](#) (object sender, RoutedEventArgs e)  
*Clears everything from map and disposes it. Then makes new window and makes the same map to it. Disables button and adds eventhandler for closing window that it can be put back to same place.*
- void [CenterMapButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Centers map to what ever listbox Assignment is selected or none if not selected.*
- void [connection\\_AssignmentUpdatedEvent](#) (ObservableCollection< [Assignment](#) > newConnections)

*Replaces assignments with a new assignment if there is one already with same guid. If there is not assignment with the guid, adds new assignment to assignments. If location is set, adds pushpin. If locationtype is set to userspecified takes old location from previous assignment and adds its pushpin so it disappear. Also displays textmessages if there is any.*

- void [AssignmentsListBox\\_SelectionChanged](#) (object sender, SelectionChangedEventArgs e)  
*If no assignment is selected -> returns. If user is handling a assignment sets assignmentlistbox selection to old one and returns so user cannot change selection while assignment is active. Also centers map when assignmentslistbox selection is changed. If selecteditem has location it enables centermapbutton. Also clears chat and loads selected assignments chat messages. Also clears videoplayer and ekg canvas.*
- void [HandleAssignment\\_Click](#) (object sender, RoutedEventArgs e)  
*If there is none selected in assignmentlistbox, shows messagebox that informs user to select one. Switch HandleAssignmentButton and handleassignmentmennuitem content and click handler to vice versa. State changing happens in own thread so it won't block ui thread. Enables buttons that can now be used.*
- void [UnHandleAssignmentButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Switch HandleAssignmentButton content and click handler to vice versa. State changing happens in own thread so it won't block ui thread. Disables buttons that user cannot use when assignment is not active.*
- void [GetMobileUserLocationButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Asks connection to show map for mobile client.*
- void [GetVideoButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Switch getVideoButton content and click handler and to vice versa.*
- void [StopVideoButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Switch getVideoButton content and click handler and to vice versa.*
- void [ChangeAssignmentPriority\\_Click](#) (object sender, RoutedEventArgs e)  
*Tries first cast sender to comboboxitem and from its tag cast it to priority and then ask connection to update it. If cast to comboboxitem fails it tries to cast it to menuitem and tries same again.*
- void [RequestMeasurementDataFromDevice\\_Click](#) (object sender, RoutedEventArgs e)  
*Requests measurement data from remote device and moves to cancel request state. If ekg expander is collapsed -> expands it.*
- void [CancelMeasurementDataFromDevice\\_Click](#) (object sender, RoutedEventArgs e)  
*Request cancel from mobile client and returns to request start mode.*
- void [MeasurementDataReceived](#) (object sender, EventArgs e, [MeasurementInstrumentDto](#) instrument, byte[] measurementData, int dataSamplesPerSecond)  
*Adds points to graph from bytes array and scrolls to end of it.*
- void [ChatSendButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Only works if chattextbox is not empty and there is active assignment if not, returns. This just sends message to mobile client and then clears input box.*
- void [IncomingPictureHandler](#) (object sender, byte[] image)  
*Displays the incoming "video" pictures from the mobile emergency client in the Image named ImageElement defined in the MainWindow.xaml*
- void [ChatTextBox\\_KeyUp](#) (object sender, KeyEventArgs e)  
*If key pressed is enter clicks chats send button.*
- void [RotateVideoToLeftButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Transforms imageElement center point to origo and then rotates it 90 degrees and then transforms it back to same x,y position.*
- void [RotateVideoToRightButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Transforms imageElement center point to origo and then rotates it -90 degrees and then transforms it back to same x,y position.*
- void [SettingsMenuitem\\_Click](#) (object sender, RoutedEventArgs e)  
*Opens settings window.*
- void [MeasurementDataExpander\\_Collapsed](#) (object sender, RoutedEventArgs e)  
*Changes expander header to "avaa".*
- void [MeasurementDataExpander\\_Expanded](#) (object sender, RoutedEventArgs e)  
*Changes expander header to "pilota".*



- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)

### Private Attributes

- ObservableCollection< [Assignment](#) > [assignments](#) = new ObservableCollection<[Assignment](#)>()
- [Assignment](#) [ownAssignment](#)
- [Connection](#) [connection](#)
- [AudioVideoTransferManager](#) [audioVideoManager](#)
- bool [assignmentActive](#) = false
- [Graph](#) [measurementGraph](#)
- [Settings](#) [settings](#) = new [Settings](#)()
- int [angle](#) = 0
- [SettingsWindow](#) [settingsWindow](#)
- [MapController](#) [mapController](#) = new [MapController](#)()
- bool [isVideoPlaying](#) = false
- bool [isMeasurementPlaying](#) = false
- bool [FirstMeasurementPacket](#) = true
- bool [\\_contentLoaded](#)

### 6.22.1 Detailed Description

Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.

[MainWindow](#)

<author>Atte Söderlund</author>

### 6.22.2 Constructor & Destructor Documentation

#### 6.22.2.1 Hake\_WPF.MainWindow.MainWindow ( )

Initializes components and focuses map. Also connects to server by endpointAddress that is in settings or by default. Then adds eventhandlers and informs user if connection is not established in case there is some problem. Also starts Pinger that pings server.

### 6.22.3 Member Function Documentation

#### 6.22.3.1 void Hake\_WPF.MainWindow.AssignmentsListBox\_SelectionChanged ( object sender, SelectionChangedEventArgs e ) [private]

If no assignment is selected -> returns. If user is handling a assignment sets assignmentlistbox selection to old one and returns so user cannot change selection while assignment is active. Also centers map when assignmentslistbox selection is changed. If selecteditem has location it enables centermapbutton. Also clears chat and loads selected assignments chat messages. Also clears videoplayer and ekg canvas.

#### 6.22.3.2 void Hake\_WPF.MainWindow.CancelMeasurementDataFromDevice\_Click ( object sender, RoutedEventArgs e ) [private]

Request cancel from mobile client and returns to request start mode.

<author>Niko Mononen</author>

6.22.3.3 void Hake\_WPF.MainWindow.CenterMapButton\_Click ( object sender, RoutedEventArgs e ) [private]

Centers map to what ever listbox Assignment is selected or none if not selected.

6.22.3.4 void Hake\_WPF.MainWindow.ChangeAssignmentPriority\_Click ( object sender, RoutedEventArgs e ) [private]

Tries first cast sender to comboboxitem and from its tag cast it to priority and then ask connection to update it. If cast to comboboxitem fails it tries to cast it to menuitem and tries same again.

6.22.3.5 void Hake\_WPF.MainWindow.ChatSendButton\_Click ( object sender, RoutedEventArgs e ) [private]

Only works if chattextbox is not empty and there is active assignment if not, returns. This just sends message to mobile client and then clears input box.

6.22.3.6 void Hake\_WPF.MainWindow.ChatTextBox\_KeyUp ( object sender, KeyEventArgs e ) [private]

If key pressed is enter clicks chats send button.

6.22.3.7 void System.Windows.Markup.IComponentConnector. Hake\_WPF.MainWindow.Connect ( int connectionId, object target ) [private]

6.22.3.8 void System.Windows.Markup.IComponentConnector. Hake\_WPF.MainWindow.Connect ( int connectionId, object target ) [private]

6.22.3.9 void Hake\_WPF.MainWindow.connection\_AssignmentUpdatedEvent ( ObservableCollection< Assignment > newConnections ) [private]

Replaces assignments with a new assignment if there is one already with same guid. If there is not assignment with the guid, adds new assignment to assignments. If location is set, adds pushpin. If locationtype is set to userspecified takes old location from previous assignment and adds its puhspin so it disapear. Also displays textmessages if there is any.

#### Parameters

<i>newConnections</i>	Observablecollection of new connections
-----------------------	---

6.22.3.10 void Hake\_WPF.MainWindow.GetMobileUserLocationButton\_Click ( object sender, RoutedEventArgs e ) [private]

Asks connection to show map for mobile client.

6.22.3.11 void Hake\_WPF.MainWindow.GetVideoButton\_Click ( object sender, RoutedEventArgs e ) [private]

Switch getVideoButton content and click handler and to vice versa.

6.22.3.12 void Hake\_WPF.MainWindow.HandleAssignment\_Click ( object sender, RoutedEventArgs e ) [private]

If there is none selected in assignmentlistbox, shows messagebox that informs user to select one. Switch HandleAssignmentButton and handleassignmentmennuitem content and click handler to vice versa. State changing happens in own thread so it won't block ui thread. Enables buttons that can now be used.

6.22.3.13 void Hake\_WPF.MainWindow.IncomingPictureHandler ( object *sender*, byte[] *image* ) [private]

Displays the incoming "video" pictures from the mobile emergency client in the Image named ImageElement defined in the MainWindow.xaml

<author>Veli-Mikko Puupponen</author>

#### Parameters

<i>image</i>	Image of any common format in a byte array
--------------	--

6.22.3.14 void Hake\_WPF.MainWindow.InitializeComponent ( )

InitializeComponent

6.22.3.15 void Hake\_WPF.MainWindow.InitializeComponent ( )

InitializeComponent

6.22.3.16 void Hake\_WPF.MainWindow.mapController\_MapWindowClosingEvent ( ) [private]

Enables MapTopOwnWindowButton when [MapWindow](#) is closing.

6.22.3.17 void Hake\_WPF.MainWindow.MapToOwnWindow\_Click ( object *sender*, RoutedEventArgs *e* ) [private]

Clears everything from map and disposes it. Then makes new window and makes the same map to it. Disables button and adds eventhandler for closing window that it can be put back to same place.

6.22.3.18 void Hake\_WPF.MainWindow.MeasurementDataExpander\_Collapsed ( object *sender*, RoutedEventArgs *e* ) [private]

Changes expander header to "avaa".

6.22.3.19 void Hake\_WPF.MainWindow.MeasurementDataExpander\_Expanded ( object *sender*, RoutedEventArgs *e* ) [private]

Changes expander header to "piilota".

6.22.3.20 void Hake\_WPF.MainWindow.MeasurementDataReceived ( object *sender*, EventArgs *e*, MeasurementInstrumentDto *instrument*, byte[] *measurementData*, int *dataSamplesPerSecond* ) [private]

Adds points to graph from bytes array and scrolls to end of it.

<author>Niko Mononen</author>

#### Parameters

<i>instrument</i>	Where we want measurementdata
<i>measurementData</i>	The data

<i>dataSamples↔ PerSecond</i>	Samples per second
-----------------------------------	--------------------

**6.22.3.21** `override void Hake_WPF.MainWindow.OnClosing ( System.ComponentModel.CancelEventArgs e )`  
[protected]

Disconnects before closing program. Also if mapwindows or/and settingswindow is open, closes those too.

**6.22.3.22** `void Hake_WPF.MainWindow.RequestMeasurementDataFromDevice_Click ( object sender, RoutedEventArgs e )`  
[private]

Requests measurement data from remote device and moves to cancel request state. If ekg expander is collapsed -> expands it.

<author>Niko Mononen</author>

**6.22.3.23** `void Hake_WPF.MainWindow.RotateVideoToLeftButton_Click ( object sender, RoutedEventArgs e )` [private]

Transforms imageElemnt center point to origo and then rotates it 90 degrees and then transforms it back to same x,y position.

**6.22.3.24** `void Hake_WPF.MainWindow.RotateVideoToRightButton_Click ( object sender, RoutedEventArgs e )`  
[private]

Transforms imageElemnt center point to origo and then rotates it -90 degrees and then transforms it back to same x,y position.

**6.22.3.25** `void Hake_WPF.MainWindow.SettingsMenuItem_Click ( object sender, RoutedEventArgs e )` [private]

Opens settings window.

**6.22.3.26** `void Hake_WPF.MainWindow.StopVideoButton_Click ( object sender, RoutedEventArgs e )` [private]

Switch getVideoButton content and click handler and to vice versa.

**6.22.3.27** `void Hake_WPF.MainWindow.UnHandleAssignmentButton_Click ( object sender, RoutedEventArgs e )`  
[private]

Switch HandleAssignmentButton content and click handler to vice versa. State changing happens in own thread so it won't block ui thread. Disables buttons that user cannot use when assingment is not active.

## 6.22.4 Member Data Documentation

**6.22.4.1** `bool Hake_WPF.MainWindow._contentLoaded` [private]

**6.22.4.2** `int Hake_WPF.MainWindow.angle = 0` [private]

**6.22.4.3** `bool Hake_WPF.MainWindow.assignmentActive = false` [private]

**6.22.4.4** `ObservableCollection<Assignment> Hake_WPF.MainWindow.assignments = new  
ObservableCollection<Assignment>() [private]`

- 6.22.4.5 **AudioVideoTransferManager** Hake\_WPF.MainWindow.audioVideoManager [private]
- 6.22.4.6 **Connection** Hake\_WPF.MainWindow.connection [private]
- 6.22.4.7 **bool** Hake\_WPF.MainWindow.FirstMeasurementPacket = true [private]
- 6.22.4.8 **bool** Hake\_WPF.MainWindow.isMeasurementPlaying = false [private]
- 6.22.4.9 **bool** Hake\_WPF.MainWindow.isVideoPlaying = false [private]
- 6.22.4.10 **MapController** Hake\_WPF.MainWindow.mapController = new MapController() [private]
- 6.22.4.11 **Graph** Hake\_WPF.MainWindow.measurementGraph [private]
- 6.22.4.12 **Assignment** Hake\_WPF.MainWindow.ownAssignment [private]
- 6.22.4.13 **Settings** Hake\_WPF.MainWindow.settings = new Settings() [private]
- 6.22.4.14 **SettingsWindow** Hake\_WPF.MainWindow.settingsWindow [private]

The documentation for this class was generated from the following files:

- [MainWindow.xaml.cs](#)
- [MainWindow.g.cs](#)
- [MainWindow.g.i.cs](#)

## 6.23 Hake\_WPF.MapController Class Reference

Controls maps. User can add specific pushpin to maps or all pushpins again. Maps can be too added. User can also center maps. Also handles closing maps.

### Public Member Functions

- delegate void [MapWindowClosing](#) ()  
*Invoked when [MapWindow](#) is closing.*
- [MapController](#) ()  
*Constructor. Does nothing.*
- void [AddMap](#) (Map map)  
*Adds map given map.*
- void [MapToOwnWindow](#) (ObservableCollection< [Assignment](#) > assignments)  
*Makes new mapwindow and adds that map. Then adds pushpins to maps and adds eventhandler for mapwindow closing.*
- void [Close](#) ()  
*Closes mapwindow if it is open.*
- void [AddAllPushpins](#) (ObservableCollection< [Assignment](#) > assignments)  
*Adds puhspin to all maps.*
- void [CenterMaps](#) ([Assignment](#) assignment)  
*Centers all maps to assignmets location.*
- void [AddPushPinToAssignment](#) ([Assignment](#) assignment)  
*Adds all puhspins from specific assingment.*

## Events

- [MapWindowClosing](#) [MapWindowClosingEvent](#)

## Private Member Functions

- void [AddPushPinsToMap](#) (Map map, ObservableCollection< [Assignment](#) > collection)  
*Removes all pushpins from given map and then adds pushpins from given collection to the map.*
- void [mapWindow\\_Closing](#) (object sender, System.ComponentModel.CancelEventArgs e)  
*Removes map and then removes event handler for mapwindowclosing.*

## Private Attributes

- [MapWindow](#) [mapWindow](#)
- ObservableCollection< Pushpin > [pushpins](#) = new ObservableCollection<Pushpin>()
- ObservableCollection< Map > [maps](#) = new ObservableCollection<Map>()

### 6.23.1 Detailed Description

Controls maps. User can add specific pushpin to maps or all pushpins again. Maps can be too added. User can also center maps. Also handles closing maps.

<author>Atte Söderlund</author>

### 6.23.2 Constructor & Destructor Documentation

#### 6.23.2.1 [Hake\\_WPF.MapController.MapController](#) ( )

Constructor. Does nothing.

### 6.23.3 Member Function Documentation

#### 6.23.3.1 void [Hake\\_WPF.MapController.AddAllPushpins](#) ( ObservableCollection< [Assignment](#) > *assignments* )

Adds pushpin to all maps.

Parameters

<i>assignments</i>	List of assignments that contains those pushpins
--------------------	--

#### 6.23.3.2 void [Hake\\_WPF.MapController.AddMap](#) ( Map *map* )

Adds map given map.

Parameters

<i>map</i>	Given map
------------	-----------

#### 6.23.3.3 void [Hake\\_WPF.MapController.AddPushPinsToMap](#) ( Map *map*, ObservableCollection< [Assignment](#) > *collection* ) [private]

Removes all pushpins from given map and then adds pushpins from given collection to the map.

## Parameters

<i>map</i>	Where you want the pushpins
<i>collection</i>	Collection that contains Assignments

6.23.3.4 void Hake\_WPF.MapController.AddPushPinToAssignment ( Assignment *assignment* )

Adds all puhspins from specific assingment.

## Parameters

<i>assignment</i>	Specific assingment that puhspins is added
-------------------	--

6.23.3.5 void Hake\_WPF.MapController.CenterMaps ( Assignment *assignment* )

Centers all maps to assignmets location.

## Parameters

<i>assignment</i>	<a href="#">Assignment</a> that contains location
-------------------	---

## 6.23.3.6 void Hake\_WPF.MapController.Close ( )

Closes mapwindow if it is open.

6.23.3.7 void Hake\_WPF.MapController.MapToOwnWindow ( ObservableCollection< Assignment > *assignments* )

Makes new mapwindow and adds that map. Then adds pushpins to maps and adds eventhandler for mapwindow closing.

## Parameters

<i>assignments</i>	List of assignments so puhspins can be added
--------------------	--

6.23.3.8 void Hake\_WPF.MapController.mapWindow\_Closing ( object *sender*, System.ComponentModel.CancelEventArgs *e* )  
[private]

Removes map and then removes event handler for mapwindowclosing.

## 6.23.3.9 delegate void Hake\_WPF.MapController.MapWindowClosing ( )

Invoked when [MapWindow](#) is closing.

## 6.23.4 Member Data Documentation

## 6.23.4.1 ObservableCollection&lt;Map&gt; Hake\_WPF.MapController.maps = new ObservableCollection&lt;Map&gt;() [private]

## 6.23.4.2 MapWindow Hake\_WPF.MapController.mapWindow [private]

6.23.4.3 ObservableCollection<Pushpin> Hake\_WPF.MapController.pushpins = new ObservableCollection<Pushpin>()  
[private]

### 6.23.5 Event Documentation

#### 6.23.5.1 MapWindowClosing Hake\_WPF.MapController.MapWindowClosingEvent

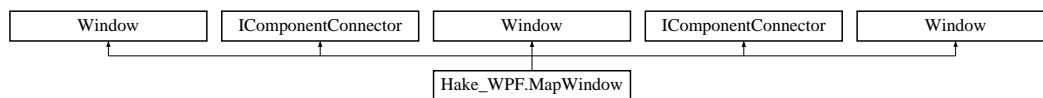
The documentation for this class was generated from the following file:

- [MapController.cs](#)

## 6.24 Hake\_WPF.MapWindow Class Reference

Simple resizable window that only contains map.

Inheritance diagram for Hake\_WPF.MapWindow:



### Public Member Functions

- [MapWindow](#) (Location location, int zoomLevel)  
*Initializes [MapWindow](#) with cetered to given location and with given zoomlevel.*
- void [InitializeComponent](#) ()  
*InitializeComponent*
- void [InitializeComponent](#) ()  
*InitializeComponent*

### Private Member Functions

- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)

### Private Attributes

- bool [\\_contentLoaded](#)

#### 6.24.1 Detailed Description

Simple resizable window that only contains map.

[MapWindow](#)

#### 6.24.2 Constructor & Destructor Documentation

##### 6.24.2.1 Hake\_WPF.MapWindow.MapWindow ( Location location, int zoomLevel )

Initializes [MapWindow](#) with cetered to given location and with given zoomlevel.



## Parameters

<i>location</i>	Location where to center map
<i>zoomLevel</i>	Wanted zoomlevel

## 6.24.3 Member Function Documentation

6.24.3.1 void System.Windows.Markup.IComponentConnector. Hake\_WPF.MapWindow.Connect ( int *connectionId*, object *target* ) [private]

6.24.3.2 void System.Windows.Markup.IComponentConnector. Hake\_WPF.MapWindow.Connect ( int *connectionId*, object *target* ) [private]

6.24.3.3 void Hake\_WPF.MapWindow.InitializeComponent ( )

InitializeComponent

6.24.3.4 void Hake\_WPF.MapWindow.InitializeComponent ( )

InitializeComponent

## 6.24.4 Member Data Documentation

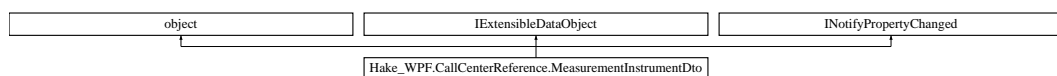
6.24.4.1 bool Hake\_WPF.MapWindow.\_contentLoaded [private]

The documentation for this class was generated from the following files:

- [MapWindow.xaml.cs](#)
- [MapWindow.g.cs](#)
- [MapWindow.g.i.cs](#)

## 6.25 Hake\_WPF.CallCenterReference.MeasurementInstrumentDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.MeasurementInstrumentDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- int [DataHeaderLength](#) [get, set]
- int [DataSampleChannels](#) [get, set]
- int [DataSampleSize](#) [get, set]
- int [DataSamplesPerSecond](#) [get, set]
- string [DeviceDescription](#) [get, set]

- string [DeviceIdentifier](#) [get, set]
- [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto DeviceType](#) [get, set]
- bool [HeaderRepeatsOnEveryPacket](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- int [DataHeaderLengthField](#)
- int [DataSampleChannelsField](#)
- int [DataSampleSizeField](#)
- int [DataSamplesPerSecondField](#)
- string [DeviceDescriptionField](#)
- string [DeviceIdentifierField](#)
- [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto DeviceTypeField](#)
- bool [HeaderRepeatsOnEveryPacketField](#)

## 6.25.1 Member Function Documentation

- 6.25.1.1 void [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.RaisePropertyChanged \( string propertyName \)](#)  
[protected]

## 6.25.2 Member Data Documentation

- 6.25.2.1 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataHeaderLengthField](#) [private]
- 6.25.2.2 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataSampleChannelsField](#) [private]
- 6.25.2.3 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataSampleSizeField](#) [private]
- 6.25.2.4 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataSamplesPerSecondField](#) [private]
- 6.25.2.5 string [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceDescriptionField](#) [private]
- 6.25.2.6 string [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceIdentifierField](#) [private]
- 6.25.2.7 [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceTypeField](#) [private]↔
- 6.25.2.8 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.extensionDataField](#) [private]↔
- 6.25.2.9 bool [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.HeaderRepeatsOnEveryPacketField](#) [private]

## 6.25.3 Property Documentation

- 6.25.3.1 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataHeaderLength](#) [get], [set]
- 6.25.3.2 int [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto.DataSampleChannels](#) [get], [set]

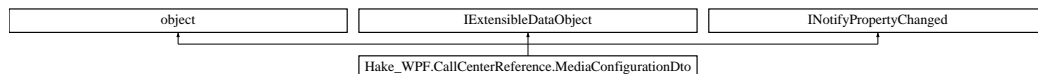
- 6.25.3.3 `int Hake_WPF.CallCenterReference.MeasurementInstrumentDto.DataSampleSize` `[get], [set]`
- 6.25.3.4 `int Hake_WPF.CallCenterReference.MeasurementInstrumentDto.DataSamplesPerSecond` `[get], [set]`
- 6.25.3.5 `string Hake_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceDescription` `[get], [set]`
- 6.25.3.6 `string Hake_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceIdentifier` `[get], [set]`
- 6.25.3.7 `Hake_WPF.CallCenterReference.MeasurementInstrumentTypeDto Hake_WPF.CallCenterReference.MeasurementInstrumentDto.DeviceType` `[get], [set]`
- 6.25.3.8 `System.Runtime.Serialization.ExtensionDataObject Hake_WPF.CallCenterReference.MeasurementInstrumentDto.ExtensionData` `[get], [set]`
- 6.25.3.9 `bool Hake_WPF.CallCenterReference.MeasurementInstrumentDto.HeaderRepeatsOnEveryPacket` `[get], [set]`
- 6.25.4 **Event Documentation**
- 6.25.4.1 `System.ComponentModel.PropertyChangedEventHandler Hake_WPF.CallCenterReference.MeasurementInstrumentDto.PropertyChanged`

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.26 Hake\_WPF.CallCenterReference.MediaConfigurationDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.MediaConfigurationDto:



### Protected Member Functions

- `void RaisePropertyChanged` (string propertyName)

### Properties

- `System.Runtime.Serialization.ExtensionDataObject ExtensionData` `[get, set]`
- `int AudioCompressionQuality` `[get, set]`
- `int AudioCompressionQualityMax` `[get, set]`
- `int AudioCompressionQualityMin` `[get, set]`
- `bool EnableAudio` `[get, set]`
- `bool EnablePicture` `[get, set]`
- `int PictureCompressionQuality` `[get, set]`
- `int PictureCompressionQualityMax` `[get, set]`
- `int PictureCompressionQualityMin` `[get, set]`
- `float PictureFps` `[get, set]`
- `int PictureResolution` `[get, set]`

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- int [AudioCompressionQualityField](#)
- int [AudioCompressionQualityMaxField](#)
- int [AudioCompressionQualityMinField](#)
- bool [EnableAudioField](#)
- bool [EnablePictureField](#)
- int [PictureCompressionQualityField](#)
- int [PictureCompressionQualityMaxField](#)
- int [PictureCompressionQualityMinField](#)
- float [PictureFpsField](#)
- int [PictureResolutionField](#)

### 6.26.1 Member Function Documentation

6.26.1.1 void `Hake_WPF.CallCenterReference.MediaConfigurationDto.RaisePropertyChanged ( string propertyName )`  
[protected]

### 6.26.2 Member Data Documentation

6.26.2.1 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQualityField` [private]

6.26.2.2 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQualityMaxField` [private]

6.26.2.3 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQualityMinField` [private]

6.26.2.4 bool `Hake_WPF.CallCenterReference.MediaConfigurationDto.EnableAudioField` [private]

6.26.2.5 bool `Hake_WPF.CallCenterReference.MediaConfigurationDto.EnablePictureField` [private]

6.26.2.6 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.MediaConfigurationDto.↔ extensionDataField` [private]

6.26.2.7 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQualityField` [private]

6.26.2.8 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQualityMaxField` [private]

6.26.2.9 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQualityMinField` [private]

6.26.2.10 float `Hake_WPF.CallCenterReference.MediaConfigurationDto.PictureFpsField` [private]

6.26.2.11 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.PictureResolutionField` [private]

### 6.26.3 Property Documentation

6.26.3.1 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQuality` [get], [set]

6.26.3.2 int `Hake_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQualityMax` [get], [set]

- 6.26.3.3 int Hake\_WPF.CallCenterReference.MediaConfigurationDto.AudioCompressionQualityMin [get], [set]
- 6.26.3.4 bool Hake\_WPF.CallCenterReference.MediaConfigurationDto.EnableAudio [get], [set]
- 6.26.3.5 bool Hake\_WPF.CallCenterReference.MediaConfigurationDto.EnablePicture [get], [set]
- 6.26.3.6 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.MediaConfigurationDto.  
ExtensionData [get], [set]
- 6.26.3.7 int Hake\_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQuality [get], [set]
- 6.26.3.8 int Hake\_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQualityMax [get], [set]
- 6.26.3.9 int Hake\_WPF.CallCenterReference.MediaConfigurationDto.PictureCompressionQualityMin [get], [set]
- 6.26.3.10 float Hake\_WPF.CallCenterReference.MediaConfigurationDto.PictureFps [get], [set]
- 6.26.3.11 int Hake\_WPF.CallCenterReference.MediaConfigurationDto.PictureResolution [get], [set]

## 6.26.4 Event Documentation

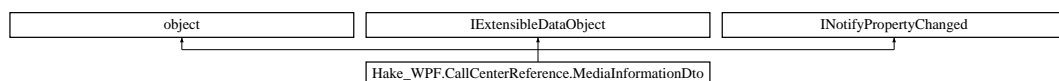
- 6.26.4.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.MediaConfigurationDto.  
PropertyChanged

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.27 Hake\_WPF.CallCenterReference.MediaInformationDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.MediaInformationDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- bool [CompressedGZip](#) [get, set]
- Hake\_WPF.CallCenterReference.MediaTypeDto [MediaType](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- bool [CompressedGZipField](#)
- [Hake\\_WPF.CallCenterReference.MediaTypeDto](#) [MediaTypeField](#)

## 6.27.1 Member Function Documentation

6.27.1.1 void [Hake\\_WPF.CallCenterReference.MedialInformationDto.RaisePropertyChanged](#) ( string *propertyName* )  
[protected]

## 6.27.2 Member Data Documentation

6.27.2.1 bool [Hake\\_WPF.CallCenterReference.MedialInformationDto.CompressedGZipField](#) [private]

6.27.2.2 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.MedialInformationDto.extensionDataField](#) [private]

6.27.2.3 [Hake\\_WPF.CallCenterReference.MediaTypeDto](#) [Hake\\_WPF.CallCenterReference.MedialInformationDto.MediaTypeField](#) [private]

## 6.27.3 Property Documentation

6.27.3.1 bool [Hake\\_WPF.CallCenterReference.MedialInformationDto.CompressedGZip](#) [get], [set]

6.27.3.2 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.MedialInformationDto.ExtensionData](#) [get], [set]

6.27.3.3 [Hake\\_WPF.CallCenterReference.MediaTypeDto](#) [Hake\\_WPF.CallCenterReference.MedialInformationDto.MediaType](#) [get], [set]

## 6.27.4 Event Documentation

6.27.4.1 System.ComponentModel.PropertyChangedEventHandler [Hake\\_WPF.CallCenterReference.MedialInformationDto.PropertyChanged](#)

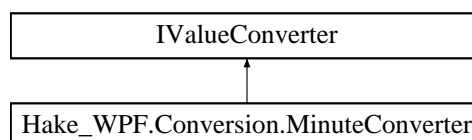
The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.28 Hake\_WPF.Conversion.MinuteConverter Class Reference

Class for converting time in minutes to a string and back.

Inheritance diagram for [Hake\\_WPF.Conversion.MinuteConverter](#):



## Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts minutes to a string in format "x h x min".*
- object [ConvertBack](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts a string of format "x h x min" to minutes.*

### 6.28.1 Detailed Description

Class for converting time in minutes to a string and back.

<author>Ilkka Rautiainen</author>

### 6.28.2 Member Function Documentation

6.28.2.1 object Hake\_WPF.Conversion.MinuteConverter.Convert ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

Converts minutes to a string in format "x h x min".

#### Parameters

<i>value</i>	value
<i>targetType</i>	targetType
<i>parameter</i>	parameter
<i>culture</i>	culture

#### Returns

string in format "x h x min"

6.28.2.2 object Hake\_WPF.Conversion.MinuteConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

Converts a string of format "x h x min" to minutes.

#### Parameters

<i>value</i>	value
<i>targetType</i>	targetType
<i>parameter</i>	parameter
<i>culture</i>	culture

#### Returns

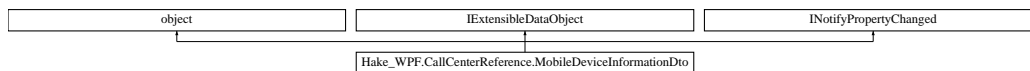
total minutes

The documentation for this class was generated from the following file:

- [MinuteConverter.cs](#)

## 6.29 Hake\_WPF.CallCenterReference.MobileDeviceInformationDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.MobileDeviceInformationDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- bool [CellularEnabled](#) [get, set]
- string [CellularMobileOperator](#) [get, set]
- bool [NetworkAvailable](#) [get, set]
- int [RemainingChargePercent](#) [get, set]
- int [RemainingDischargeTime](#) [get, set]
- bool [RoamingEnabled](#) [get, set]
- bool [WiFiEnabled](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- bool [CellularEnabledField](#)
- string [CellularMobileOperatorField](#)
- bool [NetworkAvailableField](#)
- int [RemainingChargePercentField](#)
- int [RemainingDischargeTimeField](#)
- bool [RoamingEnabledField](#)
- bool [WiFiEnabledField](#)

## 6.29.1 Member Function Documentation

- 6.29.1.1 void `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.RaisePropertyChanged ( string propertyName )` [protected]

## 6.29.2 Member Data Documentation

- 6.29.2.1 bool `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.CellularEnabledField` [private]
- 6.29.2.2 string `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.CellularMobileOperatorField` [private]
- 6.29.2.3 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.extensionDataField` [private]
- 6.29.2.4 bool `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.NetworkAvailableField` [private]
- 6.29.2.5 int `Hake_WPF.CallCenterReference.MobileDeviceInformationDto.RemainingChargePercentField` [private]



6.29.2.6 int Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.RemainingDischargeTimeField [private]

6.29.2.7 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.RoamingEnabledField [private]

6.29.2.8 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.WiFiEnabledField [private]

### 6.29.3 Property Documentation

6.29.3.1 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.CellularEnabled [get], [set]

6.29.3.2 string Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.CellularMobileOperator [get], [set]

6.29.3.3 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.  
ExtensionData [get], [set]

6.29.3.4 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.NetworkAvailable [get], [set]

6.29.3.5 int Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.RemainingChargePercent [get], [set]

6.29.3.6 int Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.RemainingDischargeTime [get], [set]

6.29.3.7 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.RoamingEnabled [get], [set]

6.29.3.8 bool Hake\_WPF.CallCenterReference.MobileDeviceInformationDto.WiFiEnabled [get], [set]

### 6.29.4 Event Documentation

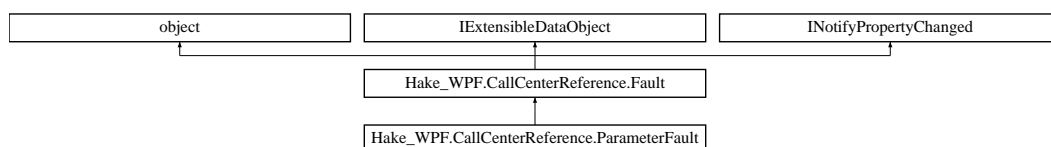
6.29.4.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.MobileDeviceInformation  
Dto.PropertyChanged

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.30 Hake\_WPF.CallCenterReference.ParameterFault Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.ParameterFault:



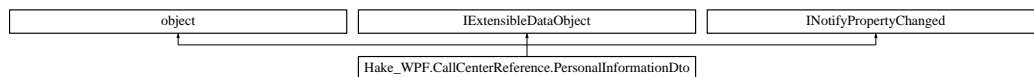
### Additional Inherited Members

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.31 Hake\_WPF.CallCenterReference.PersonalInformationDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.PersonalInformationDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [Locality](#) [get, set]
- string [Name](#) [get, set]
- string[] [PhoneNumbers](#) [get, set]
- int [PostalCode](#) [get, set]
- string [StreetAddress](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [LocalityField](#)
- string [NameField](#)
- string[] [PhoneNumbersField](#)
- int [PostalCodeField](#)
- string [StreetAddressField](#)

### 6.31.1 Member Function Documentation

6.31.1.1 void `Hake_WPF.CallCenterReference.PersonalInformationDto.RaisePropertyChanged ( string propertyName )`  
[protected]

### 6.31.2 Member Data Documentation

6.31.2.1 System.Runtime.Serialization.ExtensionDataObject `Hake_WPF.CallCenterReference.PersonalInformationDto.extensionDataField` [private]

6.31.2.2 string `Hake_WPF.CallCenterReference.PersonalInformationDto.LocalityField` [private]

6.31.2.3 string `Hake_WPF.CallCenterReference.PersonalInformationDto.NameField` [private]

6.31.2.4 string [] `Hake_WPF.CallCenterReference.PersonalInformationDto.PhoneNumbersField` [private]

6.31.2.5 int `Hake_WPF.CallCenterReference.PersonalInformationDto.PostalCodeField` [private]

6.31.2.6 string `Hake_WPF.CallCenterReference.PersonalInformationDto.StreetAddressField` [private]

### 6.31.3 Property Documentation

6.31.3.1 `System.Runtime.Serialization.ExtensionDataObject` `Hake_WPF.CallCenterReference.PersonallInformationDto`.↔  
`ExtensionData` [get], [set]

6.31.3.2 `string` `Hake_WPF.CallCenterReference.PersonallInformationDto.Locality` [get], [set]

6.31.3.3 `string` `Hake_WPF.CallCenterReference.PersonallInformationDto.Name` [get], [set]

6.31.3.4 `string []` `Hake_WPF.CallCenterReference.PersonallInformationDto.PhoneNumbers` [get], [set]

6.31.3.5 `int` `Hake_WPF.CallCenterReference.PersonallInformationDto.PostalCode` [get], [set]

6.31.3.6 `string` `Hake_WPF.CallCenterReference.PersonallInformationDto.StreetAddress` [get], [set]

## 6.31.4 Event Documentation

6.31.4.1 `System.ComponentModel.PropertyChangedEventHandler` `Hake_WPF.CallCenterReference.PersonallInformationDto`.↔  
`PropertyChanged`

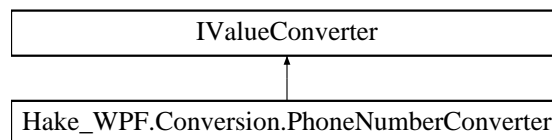
The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.32 Hake\_WPF.Conversion.PhoneNumberConverter Class Reference

Converter for location.

Inheritance diagram for `Hake_WPF.Conversion.PhoneNumberConverter`:



## Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts string[] to string so that each unit from string[] is in one string.*
- object [ConvertBack](#) (object value, Type targetType, object parameter, CultureInfo culture)

### 6.32.1 Detailed Description

Converter for location.

<author>Atte Söderlund</author>

### 6.32.2 Member Function Documentation

6.32.2.1 `object` `Hake_WPF.Conversion.PhoneNumberConverter.Convert` ( `object value`, `Type targetType`, `object parameter`, `CultureInfo culture` )

Converts string[] to string so that each unit from string[] is in one string.

## Parameters

<i>value</i>	String[] of phonenumbers
<i>targetType</i>	Not in use
<i>parameter</i>	Not in use
<i>culture</i>	Not in use

## Returns

6.32.2.2 object Hake\_WPF.Conversion.PhoneNumberConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

The documentation for this class was generated from the following file:

- [PhoneNumberConverter.cs](#)

## 6.33 Hake\_WPF.Network.Pinger Class Reference

Class for sending keepalive messages to server.

### Public Member Functions

- [Pinger](#) (int interval, [Connection connection](#))  
*Creates the pinger.*
- void [StartPinger](#) ()  
*Starts the pinger by creating a new timer and an event handler. Keepalive message interval is entered here.*

### Private Member Functions

- void [StopPinger](#) ()  
*Stops the pinger by removing the event handler.*
- async void [SendKeepalive](#) (object sender, EventArgs e)  
*Sends the keepalive message on regular intervals. If the connection is lost, stops the pinger and shows a message box to user.*

### Private Attributes

- int [keepaliveIntervalSeconds](#)
- [Connection connection](#)
- DispatcherTimer [dispatcherTimer](#)

#### 6.33.1 Detailed Description

Class for sending keepalive messages to server.

<author>Ilkka Rautiainen</author>

## 6.33.2 Constructor & Destructor Documentation

### 6.33.2.1 Hake\_WPF.Network.Pinger.Pinger ( int *interval*, Connection *connection* )

Creates the pinger.

## Parameters

<i>interval</i>	interval of keepalive messages in seconds
<i>connection</i>	connection

### 6.33.3 Member Function Documentation

6.33.3.1 `async void Hake_WPF.Network.Pinger.SendKeepalive ( object sender, EventArgs e ) [private]`

Sends the keepalive message on regular intervals. If the connection is lost, stops the pinger and shows a message box to user.

6.33.3.2 `void Hake_WPF.Network.Pinger.StartPinger ( )`

Starts the pinger by creating a new timer and an event handler. Keepalive message interval is entered here.

6.33.3.3 `void Hake_WPF.Network.Pinger.StopPinger ( ) [private]`

Stops the pinger by removing the event handler.

### 6.33.4 Member Data Documentation

6.33.4.1 `Connection Hake_WPF.Network.Pinger.connection [private]`

6.33.4.2 `DispatcherTimer Hake_WPF.Network.Pinger.dispatcherTimer [private]`

6.33.4.3 `int Hake_WPF.Network.Pinger.keepaliveIntervalSeconds [private]`

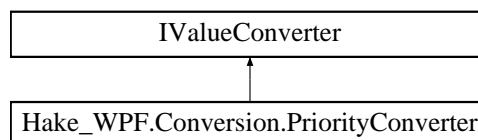
The documentation for this class was generated from the following file:

- [Pinger.cs](#)

## 6.34 Hake\_WPF.Conversion.PriorityConverter Class Reference

Converter for priority to string.

Inheritance diagram for Hake\_WPF.Conversion.PriorityConverter:



### Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts priority to kiireellinen or ei-kiireellinen.*
- object [ConvertBack](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts string back to priority.*

### 6.34.1 Detailed Description

Converter for priority to string.

<author>Atte Söderlund</author>

### 6.34.2 Member Function Documentation

#### 6.34.2.1 object Hake\_WPF.Conversion.PriorityConverter.Convert ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

Converts priority to kiireellinen or ei-kiireellinen.

##### Parameters

<i>value</i>	Priority
<i>targetType</i>	Not in use
<i>parameter</i>	Not in use
<i>culture</i>	Not in use

##### Returns

#### 6.34.2.2 object Hake\_WPF.Conversion.PriorityConverter.ConvertBack ( object *value*, Type *targetType*, object *parameter*, CultureInfo *culture* )

Converts string back to priority.

##### Parameters

<i>value</i>	String
<i>targetType</i>	Not in use
<i>parameter</i>	Not in use
<i>culture</i>	Not in use

##### Returns

The documentation for this class was generated from the following file:

- [PriorityConverter.cs](#)

## 6.35 Hake\_WPF.Settings Class Reference

User can store object that they specifie with key. User can add, update, get and remove those objects using the key.

### Public Member Functions

- [Settings](#) ()  
*initializes settings. Reads settings.cfg.*
- void [AddOrUpdate](#) (String key, object obj)  
*Adds or update object with given key.*

- void [Remove](#) (String key)  
*Removed object with given key.*
- object [Get](#) (String key)  
*Returns object from settings that is stored using given key.*
- bool [Save](#) ()  
*Save the settings to settings.cfg if its not in use.*

## Public Attributes

- const String [ENDPOINTADDRESS](#) = "endpointAddress"
- const String [NEWURGENTSTATECOLOR](#) = "newUrgentStateColor"
- const String [GRAPHLINECOLOR](#) = "graphLineColor"
- const String [GRAPHBACKGROUNDCOLOR](#) = "graphBackgroundColor"

## Private Attributes

- IsolatedStorageFile [store](#) = IsolatedStorageFile.GetUserStoreForAssembly()
- BinaryFormatter [formatter](#) = new BinaryFormatter()
- Dictionary< string, object > [settings](#) = new Dictionary<string, object>()

### 6.35.1 Detailed Description

User can store object that they specifie with key. User can add, update, get and remove those objects using the key.

<author>Atte Söderlund</author>

### 6.35.2 Constructor & Destructor Documentation

#### 6.35.2.1 Hake\_WPF.Settings.Settings ( )

initializes settings. Reads settings.cfg.

### 6.35.3 Member Function Documentation

#### 6.35.3.1 void Hake\_WPF.Settings.AddOrUpdate ( String key, object obj )

Adds or update object with given key.

Parameters

<i>key</i>	String that specifies the object
<i>obj</i>	what you want to store

#### 6.35.3.2 object Hake\_WPF.Settings.Get ( String key )

Returns object from settings that is stored using given key.

Parameters



<i>key</i>	Key string
------------	------------

**Returns**

Object that is stored

**6.35.3.3 void Hake\_WPF.Settings.Remove ( String key )**

Removed object with given key.

**Parameters**

<i>key</i>	
------------	--

**6.35.3.4 bool Hake\_WPF.Settings.Save ( )**

Save the settings to settings.cfg if its not in use.

**Returns**

True if no exceptions caught

**6.35.4 Member Data Documentation**

**6.35.4.1** `const String Hake_WPF.Settings.ENDPOINTADDRESS = "endpointAddress"`

**6.35.4.2** `BinaryFormatter Hake_WPF.Settings.formatter = new BinaryFormatter() [private]`

**6.35.4.3** `const String Hake_WPF.Settings.GRAPHBACKGROUNDCOLOR = "graphBackgroundColor"`

**6.35.4.4** `const String Hake_WPF.Settings.GRAPHLINECOLOR = "graphLineColor"`

**6.35.4.5** `const String Hake_WPF.Settings.NEWURGENTSTATECOLOR = "newUrgentStateColor"`

**6.35.4.6** `Dictionary<string, object> Hake_WPF.Settings.settings = new Dictionary<string, object>() [private]`

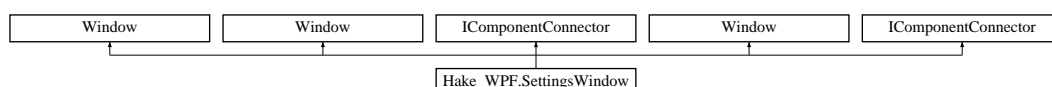
**6.35.4.7** `IsolatedStorageFile Hake_WPF.Settings.store = IsolatedStorageFile.GetUserStoreForAssembly() [private]`

The documentation for this class was generated from the following file:

- [Settings.cs](#)

**6.36 Hake\_WPF.SettingsWindow Class Reference**[SettingsWindow](#)

Inheritance diagram for Hake\_WPF.SettingsWindow:



## Public Member Functions

- void [InitializeComponent](#) ()  
*InitializeComponent*
- void [InitializeComponent](#) ()  
*InitializeComponent*
- [SettingsWindow](#) ()  
*Initializes settings. Loads example data.*

## Private Member Functions

- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void  
System.Windows.Markup.IComponentConnector. [Connect](#) (int connectionId, object target)
- void [PreviewListBox\\_SelectionChanged](#) (object sender, SelectionChangedEventArgs e)  
*Always when listbox selection is changed changes back to -1 so it is not selected.*
- void [StateComboBox\\_SelectionChanged](#) (object sender, SelectionChangedEventArgs e)  
*Adds state from bomboBox selection to settings but do not save it.*
- void [SaveButton\\_Click](#) (object sender, RoutedEventArgs e)  
*Saves settings.*

## Private Attributes

- bool [\\_contentLoaded](#)
- [Settings](#) settings = new [Settings](#)()

### 6.36.1 Detailed Description

#### [SettingsWindow](#)

Interaction logic for SettingsWindow.xaml. Just manages settings by ui.

<author>Atte Söderlund</author>

### 6.36.2 Constructor & Destructor Documentation

#### 6.36.2.1 [Hake\\_WPF.SettingsWindow.SettingsWindow](#) ( )

Initializes settings. Loads example data.

### 6.36.3 Member Function Documentation

6.36.3.1 void System.Windows.Markup.IComponentConnector. [Hake\\_WPF.SettingsWindow.Connect](#) ( int *connectionId*, object *target* ) [private]

6.36.3.2 void System.Windows.Markup.IComponentConnector. [Hake\\_WPF.SettingsWindow.Connect](#) ( int *connectionId*, object *target* ) [private]

6.36.3.3 void [Hake\\_WPF.SettingsWindow.InitializeComponent](#) ( )

[InitializeComponent](#)

6.36.3.4 void Hake\_WPF.SettingsWindow.InitializeComponent ( )

InitializeComponent

6.36.3.5 void Hake\_WPF.SettingsWindow.PreviewListBox\_SelectionChanged ( object sender, SelectionChangedEventArgs e )  
[private]

Always when listbox selection is changed changes back to -1 so it is not selected.

6.36.3.6 void Hake\_WPF.SettingsWindow.SaveButton\_Click ( object sender, RoutedEventArgs e ) [private]

Saves settings.

6.36.3.7 void Hake\_WPF.SettingsWindow.StateComboBox\_SelectionChanged ( object sender, SelectionChangedEventArgs e )  
[private]

Adds state from bomboBox selection to settings but do not save it.

## 6.36.4 Member Data Documentation

6.36.4.1 bool Hake\_WPF.SettingsWindow.\_contentLoaded [private]

6.36.4.2 Settings Hake\_WPF.SettingsWindow.settings = new Settings() [private]

The documentation for this class was generated from the following files:

- [SettingsWindow.g.cs](#)
- [SettingsWindow.g.i.cs](#)
- [SettingsWindow.xaml.cs](#)

## 6.37 Hake\_WPF.AudioVideoManagers.SpeexCompression Class Reference

Class providing static methods for compression and decompression of speex encoded audio segments.

### Static Public Member Functions

- static byte[] [DecompressSpeex](#) (SpeexDecoder decoder, byte[] data, int originatingLength)  
*Decompresses the provided MediaPacket's payload data using the provided SpeexDecoder. Returns resulting PCM samples in a byte array.*
- static int [CompressSpeex](#) (byte[] pcmSegment, int pcmByteCount, int sampleSizeBytes, SpeexEncoder encoder, out byte[] compressed)  
*Encodes the provided PCM samples using the provided SpeexEncoder. The resulting encoded bytes are put into the compressed array. Returns the count of 16 bit samples that was used as the input of the speex encoder.*

### 6.37.1 Detailed Description

Class providing static methods for compression and decompression of speex encoded audio segments.

<author>Veli-Mikko Puupponen</author>

## 6.37.2 Member Function Documentation

6.37.2.1 `static int Hake_WPF.AudioVideoManagers.SpeexCompression.CompressSpeex ( byte[] pcmSegment, int pcmByteCount, int sampleSizeBytes, SpeexEncoder encoder, out byte[] compressed ) [static]`

Encodes the provided PCM samples using the provided SpeexEncoder. The resulting encoded bytes are put into the compressed array. Returns the count of 16 bit samples that was used as the input of the speex encoder.

If the count of the PCM samples is not a multiple 320, the difference is padded with silence after the samples.

The SpeexEncoder is assumed to be operating in the BandMode.Wide and the sampleSizeBytes is assumed to be 2, i.e. 16bit PCM.

### Parameters

<i>pcmSegment</i>	16Bit PCM samples to encode into speex
<i>sampleSize↔ Bytes</i>	Size of the PCM samples in bytes, should be 2
<i>encoder</i>	SpeexEncoder used to encode the data
<i>compressed</i>	The target array for the resulting speex encoded audio

### Returns

The count of 16Bit PCM samples compressed.

6.37.2.2 `static byte [] Hake_WPF.AudioVideoManagers.SpeexCompression.DecompressSpeex ( SpeexDecoder decoder, byte[] data, int originatingLength ) [static]`

Decompresses the provided MediaPacket's payload data using the provided SpeexDecoder. Returns resulting PCM samples in a byte array.

The SpeexDecodes is assumed to operate in the BandMode.Wide and the encoded data to conform to this format.

If the decompression fails, returns an empty array.

### Parameters

<i>decoder</i>	SpeexDecodes instance used to decode the speex encoding
<i>data</i>	Speex encoded audio
<i>originating↔ Length</i>	The count of 16 bit PCM samples that were encoded to produce the provided encoded data

### Returns

Resulting PCM samples or an empty array

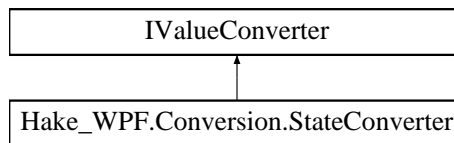
The documentation for this class was generated from the following file:

- [SpeexCompression.cs](#)

## 6.38 Hake\_WPF.Conversion.StateConverter Class Reference

Converts state to string.

Inheritance diagram for Hake\_WPF.Conversion.StateConverter:



## Public Member Functions

- object [Convert](#) (object value, Type targetType, object parameter, CultureInfo culture)  
*Converts state to string from just by searching it from states[] and using its index to cast it to state.*
- object [ConvertBack](#) (object value, Type targetType, object parameter, CultureInfo culture)

## Private Attributes

- String[] [states](#) = { "Uusi", "Käsittelyssä", "Siirrossa", "Käsitelty", "Pidossa" }

### 6.38.1 Detailed Description

Converts state to string.

<author>Atte Söderlund</author>

### 6.38.2 Member Function Documentation

6.38.2.1 object Hake\_WPF.Conversion.StateConverter.Convert ( object value, Type targetType, object parameter, CultureInfo culture )

Converts state to string from just by searching it from states[] and using its index to cast it to state.

#### Parameters

<i>value</i>	State
<i>targetType</i>	Not in use
<i>parameter</i>	Not in use
<i>culture</i>	Not in use

#### Returns

6.38.2.2 object Hake\_WPF.Conversion.StateConverter.ConvertBack ( object value, Type targetType, object parameter, CultureInfo culture )

### 6.38.3 Member Data Documentation

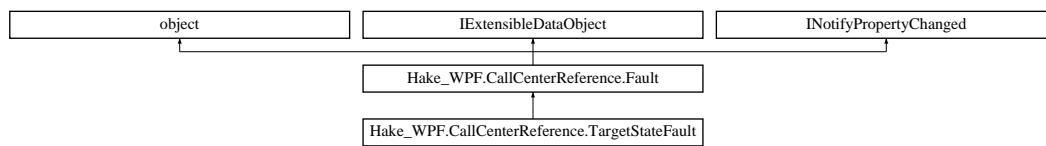
6.38.3.1 String [] Hake\_WPF.Conversion.StateConverter.states = { "Uusi", "Käsittelyssä", "Siirrossa", "Käsitelty", "Pidossa" }  
 [private]

The documentation for this class was generated from the following file:

- [StateConverter.cs](#)

### 6.39 Hake\_WPF.CallCenterReference.TargetStateFault Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.TargetStateFault:



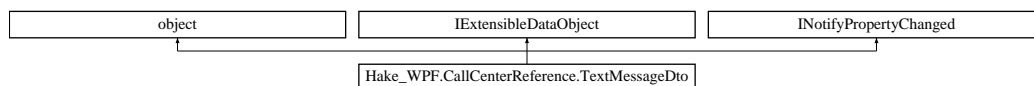
#### Additional Inherited Members

The documentation for this class was generated from the following file:

- [Reference.cs](#)

### 6.40 Hake\_WPF.CallCenterReference.TextMessageDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.TextMessageDto:



#### Public Types

- enum [MessageOriginatorDto](#) : int { [MessageOriginatorDto.MobileClient](#) = 0, [MessageOriginatorDto.CallCenterClient](#) = 1 }

#### Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

#### Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [Content](#) [get, set]
- [Hake\\_WPF.CallCenterReference.TextMessageDto.MessageOriginatorDto](#) [Originator](#) [get, set]
- System.DateTime [TimeStamp](#) [get, set]

#### Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

#### Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [ContentField](#)
- [Hake\\_WPF.CallCenterReference.TextMessageDto.MessageOriginatorDto](#) [OriginatorField](#)
- System.DateTime [TimeStampField](#)

## 6.40.1 Member Enumeration Documentation

6.40.1.1 enum Hake\_WPF.CallCenterReference.TextMessageDto.MessageOriginatorDto : int

Enumerator

***MobileClient***

***CallCenterClient***

## 6.40.2 Member Function Documentation

6.40.2.1 void Hake\_WPF.CallCenterReference.TextMessageDto.RaisePropertyChanged ( string *propertyName* )  
[protected]

## 6.40.3 Member Data Documentation

6.40.3.1 string Hake\_WPF.CallCenterReference.TextMessageDto.ContentField [private]

6.40.3.2 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.TextMessageDto.extensionData↔  
Field [private]

6.40.3.3 Hake\_WPF.CallCenterReference.TextMessageDto.MessageOriginatorDto  
Hake\_WPF.CallCenterReference.TextMessageDto.OriginatorField [private]

6.40.3.4 System.DateTime Hake\_WPF.CallCenterReference.TextMessageDto.TimeStampField [private]

## 6.40.4 Property Documentation

6.40.4.1 string Hake\_WPF.CallCenterReference.TextMessageDto.Content [get], [set]

6.40.4.2 System.Runtime.Serialization.ExtensionDataObject Hake\_WPF.CallCenterReference.TextMessageDto.ExtensionData  
[get], [set]

6.40.4.3 Hake\_WPF.CallCenterReference.TextMessageDto.MessageOriginatorDto  
Hake\_WPF.CallCenterReference.TextMessageDto.Originator [get], [set]

6.40.4.4 System.DateTime Hake\_WPF.CallCenterReference.TextMessageDto.TimeStamp [get], [set]

## 6.40.5 Event Documentation

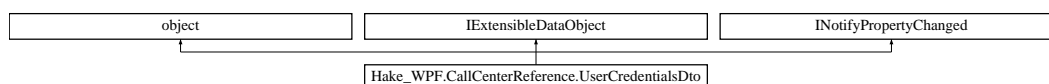
6.40.5.1 System.ComponentModel.PropertyChangedEventHandler Hake\_WPF.CallCenterReference.TextMessageDto.Property↔  
Changed

The documentation for this class was generated from the following file:

- [Reference.cs](#)

## 6.41 Hake\_WPF.CallCenterReference.UserCredentialsDto Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.UserCredentialsDto:



## Protected Member Functions

- void [RaisePropertyChanged](#) (string propertyName)

## Properties

- System.Runtime.Serialization.ExtensionDataObject [ExtensionData](#) [get, set]
- string [Password](#) [get, set]
- string [UserName](#) [get, set]

## Events

- System.ComponentModel.PropertyChangedEventHandler [PropertyChanged](#)

## Private Attributes

- System.Runtime.Serialization.ExtensionDataObject [extensionDataField](#)
- string [PasswordField](#)
- string [UserNameField](#)

### 6.41.1 Member Function Documentation

- 6.41.1.1 void [Hake\\_WPF.CallCenterReference.UserCredentialsDto.RaisePropertyChanged](#) ( string *propertyName* )  
[protected]

### 6.41.2 Member Data Documentation

- 6.41.2.1 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.UserCredentialsDto.extensionDataField](#) [private]
- 6.41.2.2 string [Hake\\_WPF.CallCenterReference.UserCredentialsDto.PasswordField](#) [private]
- 6.41.2.3 string [Hake\\_WPF.CallCenterReference.UserCredentialsDto.UserNameField](#) [private]

### 6.41.3 Property Documentation

- 6.41.3.1 System.Runtime.Serialization.ExtensionDataObject [Hake\\_WPF.CallCenterReference.UserCredentialsDto.ExtensionData](#) [get], [set]
- 6.41.3.2 string [Hake\\_WPF.CallCenterReference.UserCredentialsDto.Password](#) [get], [set]
- 6.41.3.3 string [Hake\\_WPF.CallCenterReference.UserCredentialsDto.UserName](#) [get], [set]

### 6.41.4 Event Documentation

- 6.41.4.1 System.ComponentModel.PropertyChangedEventHandler [Hake\\_WPF.CallCenterReference.UserCredentialsDto.PropertyChanged](#)

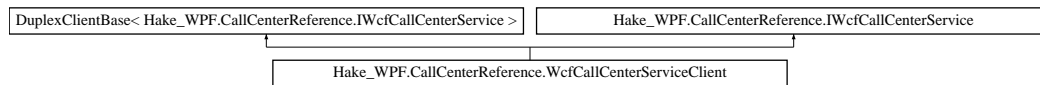
The documentation for this class was generated from the following file:

- [Reference.cs](#)



## 6.42 Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient Class Reference

Inheritance diagram for Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient:



### Public Member Functions

- [WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext callbackInstance)
- [WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext callbackInstance, string endpoint↵ ConfigurationName)
- [WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext callbackInstance, string endpoint↵ ConfigurationName, string remoteAddress)
- [WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext callbackInstance, string endpoint↵ ConfigurationName, System.ServiceModel.EndpointAddress remoteAddress)
- [WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext callbackInstance, System.Service↵ Model.Channels.Binding binding, System.ServiceModel.EndpointAddress remoteAddress)
- [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto Connect](#) (Hake\_WPF.CallCenterReference.↵ UserCredentialsDto credentials)
- System.Threading.Tasks.Task< [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) > [ConnectAsync](#) (Hake\_WPF.CallCenter↵ Reference.UserCredentialsDto credentials)
- void [Reconnect](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user)
- System.Threading.Tasks.Task [ReconnectAsync](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user)
- void [Disconnect](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user)
- System.Threading.Tasks.Task [DisconnectAsync](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user)
- [Hake\\_WPF.CallCenterReference.ConnectionDto\[\] GetActiveConnections](#) (Hake\_WPF.CallCenterReference.↵ CallCenterConnectionDto user)
- System.Threading.Tasks.Task< [Hake\\_WPF.CallCenterReference.ConnectionDto\[\]](#) > [GetActiveConnectionsAsync](#) (Hake\_WPF.Call↵ CenterReference.CallCenterConnectionDto user)
- void [OpenConnectionForProcessing](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection)
- System.Threading.Tasks.Task [OpenConnectionForProcessingAsync](#) (Hake\_WPF.CallCenterReference.↵ CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection)
- void [TransferConnection](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.Call↵ CenterReference.CallCenterConnectionDto[] targetCallCenterConnections)
- System.Threading.Tasks.Task [TransferConnectionAsync](#) (Hake\_WPF.CallCenterReference.CallCenter↵ ConnectionDto user, Hake\_WPF.CallCenterReference.CallCenterConnectionDto[] targetCallCenter↵ Connections)
- void [SetConnectionPriority](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.↵ CallCenterReference.ConnectionDto connection)
- System.Threading.Tasks.Task [SetConnectionPriorityAsync](#) (Hake\_WPF.CallCenterReference.CallCenter↵ ConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection)
- void [MoveConnectionToHold](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WP↵ F.CallCenterReference.ConnectionDto connection)
- System.Threading.Tasks.Task [MoveConnectionToHoldAsync](#) (Hake\_WPF.CallCenterReference.CallCenter↵ ConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection)
- void [MarkProcessedCloseConnection](#) (Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection)

- System.Threading.Tasks.Task [MarkProcessedCloseConnectionAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection)
- void [RequestRemoteAction](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.RemoteActionDto](#) action)
- System.Threading.Tasks.Task [RequestRemoteActionAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.RemoteActionDto](#) action)
- void [RequestMediaUpstreaming](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MediaConfigurationDto](#) mediaConfiguration)
- System.Threading.Tasks.Task [RequestMediaUpstreamingAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MediaConfigurationDto](#) mediaConfiguration)
- void [RequestMediaDownstreaming](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, string mediaUrl)
- System.Threading.Tasks.Task [RequestMediaDownstreamingAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, string mediaUrl)
- void [RequestStartMeasurement](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- System.Threading.Tasks.Task [RequestStartMeasurementAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- void [RequestStopMeasurement](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- System.Threading.Tasks.Task [RequestStopMeasurementAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#) measurementDevice)
- void [SendTextMessage](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.TextMessageDto](#) textMessage)
- System.Threading.Tasks.Task [SendTextMessageAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.ConnectionDto](#) connection, [Hake\\_WPF.CallCenterReference.TextMessageDto](#) textMessage)
- bool [UploadMediaSegment](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.MediaInformationDto](#) mediaInfo, byte[] mediaData)
- System.Threading.Tasks.Task< bool > [UploadMediaSegmentAsync](#) ([Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#) user, [Hake\\_WPF.CallCenterReference.MediaInformationDto](#) mediaInfo, byte[] mediaData)
- int [Ping](#) (int pingSequence)
- System.Threading.Tasks.Task< int > [PingAsync](#) (int pingSequence)

## 6.42.1 Constructor & Destructor Documentation

- 6.42.1.1 [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient.WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext *callbackInstance* )
- 6.42.1.2 [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient.WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext *callbackInstance*, string *endpointConfigurationName* )
- 6.42.1.3 [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient.WcfCallCenterServiceClient](#) (System.ServiceModel.InstanceContext *callbackInstance*, string *endpointConfigurationName*, string *remoteAddress* )

6.42.1.4 `Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.WcfCallCenterServiceClient ( System.ServiceModel.InstanceContext callbackInstance, string endpointConfigurationName, System.ServiceModel.EndpointAddress remoteAddress )`

6.42.1.5 `Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.WcfCallCenterServiceClient ( System.ServiceModel.InstanceContext callbackInstance, System.ServiceModel.Channels.Binding binding, System.ServiceModel.EndpointAddress remoteAddress )`

## 6.42.2 Member Function Documentation

6.42.2.1 `Hake_WPF.CallCenterReference.CallCenterConnectionDto Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.Connect ( Hake_WPF.CallCenterReference.UserCredentialsDto credentials )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.2 `System.Threading.Tasks.Task<Hake_WPF.CallCenterReference.CallCenterConnectionDto> Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.ConnectAsync ( Hake_WPF.CallCenterReference.UserCredentialsDto credentials )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.3 `void Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.Disconnect ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.4 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.DisconnectAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.5 `Hake_WPF.CallCenterReference.ConnectionDto [] Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.GetActiveConnections ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.6 `System.Threading.Tasks.Task<Hake_WPF.CallCenterReference.ConnectionDto[]> Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.GetActiveConnectionsAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.7 `void Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.MarkProcessedCloseConnection ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.8 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.MarkProcessed↔  
CloseConnectionAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.9 `void Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.MoveConnectionToHold ( Hake_WPF.Call↔  
CenterReference.CallCenterConnectionDto user, Hake_WPF.CallCenterReference.ConnectionDto  
connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.10 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.Move↔  
ConnectionToHoldAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.11 `void Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.OpenConnectionFor↔  
Processing ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.12 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.OpenConnection↔  
ForProcessingAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,  
Hake_WPF.CallCenterReference.ConnectionDto connection )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.13 `int Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.Ping ( int pingSequence )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.14 `System.Threading.Tasks.Task<int> Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.PingAsync ( int  
pingSequence )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.15 `void Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.Reconnect ( Hake_WPF.CallCenter↔  
Reference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.16 `System.Threading.Tasks.Task Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.ReconnectAsync (  
Hake_WPF.CallCenterReference.CallCenterConnectionDto user )`

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.17 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestMediaDownstreaming ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, string mediaUrl )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.18 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestMediaDownstreamingAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, string mediaUrl )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.19 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestMediaUpstreaming ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MediaConfigurationDto mediaConfiguration )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.20 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestMediaUpstreamingAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MediaConfigurationDto mediaConfiguration )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.21 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestRemoteAction ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.RemoteActionDto action )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.22 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestRemoteActionAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.RemoteActionDto action )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.23 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestStartMeasurement ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.24 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestStartMeasurementAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.25 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestStopMeasurement ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.26 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.RequestStopMeasurementAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.MeasurementInstrumentDto measurementDevice )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.27 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.SendTextMessage ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.TextMessageDto textMessage )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.28 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.SendTextMessageAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection, Hake\_WPF.CallCenterReference.TextMessageDto textMessage )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.29 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.SetConnectionPriority ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.30 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.SetConnectionPriorityAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.ConnectionDto connection )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.31 void Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.TransferConnection ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.CallCenterConnectionDto[] targetCallCenterConnections )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

6.42.2.32 System.Threading.Tasks.Task Hake\_WPF.CallCenterReference.WcfCallCenterServiceClient.TransferConnectionAsync ( Hake\_WPF.CallCenterReference.CallCenterConnectionDto user, Hake\_WPF.CallCenterReference.CallCenterConnectionDto[] targetCallCenterConnections )

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

```
6.42.2.33 bool Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.UploadMediaSegment
( Hake_WPF.CallCenterReference.CallCenterConnectionDto user, Hake_↵
_WPF.CallCenterReference.MedialInformationDto medialInfo, byte[] mediaData
)
```

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

```
6.42.2.34 System.Threading.Tasks.Task<bool> Hake_WPF.CallCenterReference.WcfCallCenterServiceClient.↵
UploadMediaSegmentAsync ( Hake_WPF.CallCenterReference.CallCenterConnectionDto user,
Hake_WPF.CallCenterReference.MedialInformationDto medialInfo, byte[] mediaData )
```

Implements [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#).

The documentation for this class was generated from the following file:

- [Reference.cs](#)





## Chapter 7

# File Documentation

### 7.1 AddressConverter.cs File Reference

#### Classes

- class [Hake\\_WPF.Conversion.AddressConverter](#)  
*Converts PersonallInfomationDto string that it (Streetname, postalcode, locality). Converback is NOT implemented and returns always null!*

#### Namespaces

- package [Hake\\_WPF.Conversion](#)

### 7.2 App.g.cs File Reference

#### Classes

- class [Hake\\_WPF.App](#)  
*Interaction logic for App.xaml*

#### Namespaces

- package [Hake\\_WPF](#)

### 7.3 App.g.i.cs File Reference

#### Classes

- class [Hake\\_WPF.App](#)  
*Interaction logic for App.xaml*

#### Namespaces

- package [Hake\\_WPF](#)

## 7.4 App.xaml.cs File Reference

### Classes

- class [Hake\\_WPF.App](#)  
*Interaction logic for App.xaml*

### Namespaces

- package [Hake\\_WPF](#)

## 7.5 AssemblyInfo.cs File Reference

## 7.6 Assignment.cs File Reference

### Classes

- class [Hake\\_WPF.Assignment](#)  
*Assignment listboxitem that contains many properties like handlers, location, time, state, priority, pushpins, assignment info and street name. It sets content and background depending state and priority and if there is location available.*

### Namespaces

- package [Hake\\_WPF](#)

## 7.7 AudioVideoTransferManager.cs File Reference

### Classes

- class [Hake\\_WPF.AudioVideoManagers.AudioVideoTransferManager](#)  
*Class for processing incoming and outgoing audio and images. Handles media data received from the server. Reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.*

### Namespaces

- package [Hake\\_WPF.AudioVideoManagers](#)

### Functions

- delegate void [Hake\\_WPF.AudioVideoManagers.JpgImageReceived](#) (object sender, byte[] imageData)  
*Delegate for image receive events.*

## 7.8 BoolConverter.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.BoolConverter](#)  
*Converts bool to string. True=on and false=ei.*

## Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.9 BufferWaveStream.cs File Reference

### Classes

- class [Hake\\_WPF.AudioVideoManagers.BufferWaveStream](#)

*Provides a NAudio WaveStream with infinite length to facilitate streaming audio playback. Contains an internal buffer for the audio samples. If the buffer is empty, all read requests will return the desired length of silence. If there is pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length.*

## Namespaces

- package [Hake\\_WPF.AudioVideoManagers](#)

## 7.10 CompressionHelper.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.CompressionHelper](#)

*Class for static compression methods.*

## Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.11 Connection.cs File Reference

### Classes

- class [Hake\\_WPF.AsyncReceiver](#)

*Receives data from server which invokes event and writes it to debug.*

- class [Hake\\_WPF.Connection](#)

*Handles communication between server and this client. User must use connect method to connect server. After that user can use rest of the methods to manage connection. Also user should handle most of events, but atleast connectionupdatedevent.*

## Namespaces

- package [Hake\\_WPF](#)

## Functions

- delegate void [Hake\\_WPF.ConnectionsUpdated](#) ([ConnectionDto](#)[] newConnections)

*Invoked when connection receives update.*

- delegate void [Hake\\_WPF.TcpMediaDataReceived](#) (object sender, [MediaInformationDto](#) info, byte[] data)

*Invoked when TcpMedia is received.*

- delegate void [Hake\\_WPF.UdpMediaDataReceived](#) (object sender, MediaPacket media)

*Invoked when udp media is received.*

- delegate void [Hake\\_WPF.MeasurementDataReceivedDelegate](#) (object sender, EventArgs e, [Measurement<InstrumentDto>](#) instrument, byte[] measurementData, int dataSamplesPerSecond)

## 7.12 GeneratedInternalTypeHelper.g.cs File Reference

### Classes

- class [XamlGeneratedNamespace.GeneratedInternalTypeHelper](#)  
[GeneratedInternalTypeHelper](#)

### Namespaces

- package [XamlGeneratedNamespace](#)

## 7.13 GeneratedInternalTypeHelper.g.i.cs File Reference

### Classes

- class [XamlGeneratedNamespace.GeneratedInternalTypeHelper](#)  
[GeneratedInternalTypeHelper](#)

### Namespaces

- package [XamlGeneratedNamespace](#)

## 7.14 GraphClass.cs File Reference

### Classes

- class [GraphClass.Graph](#)  
*Adds graph to given grid. user can add points, set bytespersecond and scroll to end.*

### Namespaces

- package [GraphClass](#)

## 7.15 LocationConverter.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.LocationConverter](#)  
*Converts location to string*

## Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.16 MainWindow.g.cs File Reference

### Classes

- class [Hake\\_WPF.MainWindow](#)  
*Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.17 MainWindow.g.i.cs File Reference

### Classes

- class [Hake\\_WPF.MainWindow](#)  
*Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.18 MainWindow.xaml.cs File Reference

### Classes

- class [Hake\\_WPF.MainWindow](#)  
*Interaction logic for MainWindow.xaml. User can ask information from mobile client: video and location. Or user can send tutorials. Also user can manage assignments here that are shown in listbox and in map by pushpins. All controls is here in mainwindow.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.19 MapController.cs File Reference

### Classes

- class [Hake\\_WPF.MapController](#)  
*Controls maps. User can add specific pushpin to maps or all pushpins again. Maps can be too added. User can also center maps. Also handles closing maps.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.20 MapWindow.g.cs File Reference

### Classes

- class [Hake\\_WPF.MapWindow](#)  
*Simple resizable window that only contains map.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.21 MapWindow.g.i.cs File Reference

### Classes

- class [Hake\\_WPF.MapWindow](#)  
*Simple resizable window that only contains map.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.22 MapWindow.xaml.cs File Reference

### Classes

- class [Hake\\_WPF.MapWindow](#)  
*Simple resizable window that only contains map.*

## Namespaces

- package [Hake\\_WPF](#)

## 7.23 MinuteConverter.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.MinuteConverter](#)  
*Class for converting time in minutes to a string and back.*

## Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.24 PhoneNumberConverter.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.PhoneNumberConverter](#)  
*Converter for location.*

### Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.25 Pinger.cs File Reference

### Classes

- class [Hake\\_WPF.Network.Pinger](#)  
*Class for sending keepalive messages to server.*

### Namespaces

- package [Hake\\_WPF.Network](#)

## 7.26 PriorityConverter.cs File Reference

### Classes

- class [Hake\\_WPF.Conversion.PriorityConverter](#)  
*Converter for priority to string.*

### Namespaces

- package [Hake\\_WPF.Conversion](#)

## 7.27 Reference.cs File Reference

### Classes

- class [Hake\\_WPF.CallCenterReference.UserCredentialsDto](#)
- class [Hake\\_WPF.CallCenterReference.CallCenterConnectionDto](#)
- class [Hake\\_WPF.CallCenterReference.Fault](#)
- class [Hake\\_WPF.CallCenterReference.TargetStateFault](#)
- class [Hake\\_WPF.CallCenterReference.ParameterFault](#)
- class [Hake\\_WPF.CallCenterReference.ConnectionFault](#)
- class [Hake\\_WPF.CallCenterReference.ConnectionDto](#)
- class [Hake\\_WPF.CallCenterReference.EmergencyTypeDto](#)
- class [Hake\\_WPF.CallCenterReference.PersonalInformationDto](#)
- class [Hake\\_WPF.CallCenterReference.LocationInformationDto](#)
- class [Hake\\_WPF.CallCenterReference.MeasurementInstrumentDto](#)
- class [Hake\\_WPF.CallCenterReference.MobileDeviceInformationDto](#)

- class [Hake\\_WPF.CallCenterReference.TextMessageDto](#)
- class [Hake\\_WPF.CallCenterReference.MediaConfigurationDto](#)
- class [Hake\\_WPF.CallCenterReference.MediaInformationDto](#)
- interface [Hake\\_WPF.CallCenterReference.IWcfCallCenterService](#)
- interface [Hake\\_WPF.CallCenterReference.IWcfCallCenterServiceCallback](#)
- interface [Hake\\_WPF.CallCenterReference.IWcfCallCenterServiceChannel](#)
- class [Hake\\_WPF.CallCenterReference.WcfCallCenterServiceClient](#)

## Namespaces

- package [Hake\\_WPF.CallCenterReference](#)

## Enumerations

- enum [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto](#) : int { [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto.NotUrgent](#) = 0, [Hake\\_WPF.CallCenterReference.ConnectionPriorityDto.Urgent](#) = 1 }
- enum [Hake\\_WPF.CallCenterReference.ConnectionStateDto](#) : int { [Hake\\_WPF.CallCenterReference.ConnectionStateDto.Arrived](#) = 0, [Hake\\_WPF.CallCenterReference.ConnectionStateDto.InProgress](#) = 1, [Hake\\_WPF.CallCenterReference.ConnectionStateDto.InTransfer](#) = 2, [Hake\\_WPF.CallCenterReference.ConnectionStateDto.Processed](#) = 3, [Hake\\_WPF.CallCenterReference.ConnectionStateDto.Hold](#) = 4 }
- enum [Hake\\_WPF.CallCenterReference.LocationTypeDto](#) : int { [Hake\\_WPF.CallCenterReference.LocationTypeDto.Initial](#) = 0, [Hake\\_WPF.CallCenterReference.LocationTypeDto.Response](#) = 1, [Hake\\_WPF.CallCenterReference.LocationTypeDto.Movement](#) = 2, [Hake\\_WPF.CallCenterReference.LocationTypeDto.UserSpecified](#) = 3 }
- enum [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto](#) : int { [Hake\\_WPF.CallCenterReference.MeasurementInstrumentTypeDto.ECG](#) = 0 }
- enum [Hake\\_WPF.CallCenterReference.RemoteActionDto](#) : int { [Hake\\_WPF.CallCenterReference.RemoteActionDto.requestLocation](#) = 0, [Hake\\_WPF.CallCenterReference.RemoteActionDto.requestDeviceStatusinformation](#) = 1, [Hake\\_WPF.CallCenterReference.RemoteActionDto.requestShowLocationMap](#) = 2, [Hake\\_WPF.CallCenterReference.RemoteActionDto.requestInstrumentList](#) = 3 }
- enum [Hake\\_WPF.CallCenterReference.MediaTypeDto](#) : int { [Hake\\_WPF.CallCenterReference.MediaTypeDto.Wave](#) = 0, [Hake\\_WPF.CallCenterReference.MediaTypeDto.Jpeg](#) = 1, [Hake\\_WPF.CallCenterReference.MediaTypeDto.AAC](#) = 2, [Hake\\_WPF.CallCenterReference.MediaTypeDto.H264](#) = 3, [Hake\\_WPF.CallCenterReference.MediaTypeDto.MP4](#) = 4, [Hake\\_WPF.CallCenterReference.MediaTypeDto.Opus](#) = 5, [Hake\\_WPF.CallCenterReference.MediaTypeDto.Speex](#) = 6 }

## 7.28 Resources.Designer.cs File Reference

### Classes

- class **Hake\_WPF.Properties.Resources**  
*A strongly-typed resource class, for looking up localized strings, etc.*

### Namespaces

- package [Hake\\_WPF.Properties](#)



## 7.29 Settings.cs File Reference

### Classes

- class [Hake\\_WPF.Settings](#)

*User can store object that they specifie with key. User can add, update, get and remove those objects using the key.*

### Namespaces

- package [Hake\\_WPF](#)

## 7.30 Settings.Designer.cs File Reference

### Classes

- class **Hake\_WPF.Properties.Settings**

### Namespaces

- package [Hake\\_WPF.Properties](#)

## 7.31 SettingsWindow.g.cs File Reference

### Classes

- class [Hake\\_WPF.SettingsWindow](#)  
*SettingsWindow*

### Namespaces

- package [Hake\\_WPF](#)

## 7.32 SettingsWindow.g.i.cs File Reference

### Classes

- class [Hake\\_WPF.SettingsWindow](#)  
*SettingsWindow*

### Namespaces

- package [Hake\\_WPF](#)

### 7.33 SettingsWindow.xaml.cs File Reference

#### Classes

- class [Hake\\_WPF.SettingsWindow](#)  
*SettingsWindow*

#### Namespaces

- package [Hake\\_WPF](#)

### 7.34 SpeexCompression.cs File Reference

#### Classes

- class [Hake\\_WPF.AudioVideoManagers.SpeexCompression](#)  
*Class providing static methods for compression and decompression of speex encoded audio segments.*

#### Namespaces

- package [Hake\\_WPF.AudioVideoManagers](#)

### 7.35 StateConverter.cs File Reference

#### Classes

- class [Hake\\_WPF.Conversion.StateConverter](#)  
*Converts state to string.*

#### Namespaces

- package [Hake\\_WPF.Conversion](#)

### 7.36 TemporaryGeneratedFile\_036C0B5B-1481-4323-8D20-8F5ADCB23D92.cs File Reference

### 7.37 TemporaryGeneratedFile\_5937a670-0e60-4077-877b-f7221da3dda1.cs File Reference

### 7.38 TemporaryGeneratedFile\_E7A71F73-0F8D-4B9B-B56E-8E70B10BC5D3.cs File Reference