Halyri - Callcenter software

1.0

Generated by Doxygen 1.8.7

Sat Jun 21 2014 19:55:52

Contents

1 Namespace Documentation **1**

1.1 Package GraphClass . . . 1

1.2 Package Hake_WPF . . . 1

1.2.1 Function Documentation . . . 2

1.3 Package Hake_WPF.AudioVideoManagers . . . 3

1.3.1 Function Documentation . . . 3

1.4 Package Hake_WPF.Conversion . . . 4

1.5 Package Hake_WPF.Network . . . 4

1.6 Package Hake_WPF.Properties . . . 4

2 Class Documentation **4**

2.1 AddressConverter Class Reference . . . 4

2.1.1 Detailed Description . . . 5

2.1.2 Member Function Documentation . . . 5

2.2 App Class Reference . . . 5

2.2.1 Detailed Description . . . 6

2.3 Assignment Class Reference . . . 6

2.3.1 Detailed Description . . . 7

2.3.2 Member Enumeration Documentation . . . 7

2.3.3 Constructor & Destructor Documentation . . . 7

2.3.4 Member Data Documentation . . . 8

2.3.5 Property Documentation . . . 8

2.3.6 Event Documentation . . . 9

2.4 AsyncReceiver Class Reference . . . 9

2.4.1 Detailed Description . . . 10

2.4.2 Member Function Documentation . . . 10

2.4.3 Member Data Documentation . . . 10

2.4.4 Event Documentation . . . 11

2.5 AudioVideoTransferManager Class Reference . . . 11

2.5.1 Detailed Description . . . 11

2.5.2 Constructor & Destructor Documentation . . . 12

2.5.3 Member Function Documentation . . . 13

2.5.4 Member Data Documentation . . . 13

2.5.5 Property Documentation . . . 13

2.6 BoolConverter Class Reference . . . 13

2.6.1 Detailed Description . . . 13

2.6.2 Member Function Documentation . . . 14

2.7 BufferWaveStream Class Reference . . . 14

2.7.1 Detailed Description . . . 15
2.7.2 Constructor & Destructor Documentation . . . 15
2.7.3 Member Function Documentation . . . 15
2.7.4 Property Documentation . . . 16
2.8 CompressionHelper Class Reference . . . 16
2.8.1 Detailed Description . . . 16
2.8.2 Member Function Documentation . . . 16
2.9 Connection Class Reference . . . 17
2.9.1 Detailed Description . . . 18
2.9.2 Constructor & Destructor Documentation . . . 19
2.9.3 Member Function Documentation . . . 20
2.9.4 Member Data Documentation . . . 23
2.9.5 Property Documentation . . . 23
2.9.6 Event Documentation . . . 24
2.10 Graph Class Reference . . . 24
2.10.1 Detailed Description . . . 24
2.10.2 Constructor & Destructor Documentation . . . 24
2.10.3 Member Function Documentation . . . 24
2.11 LocationConverter Class Reference . . . 25
2.11.1 Detailed Description . . . 25
2.11.2 Member Function Documentation . . . 25
2.12 MainWindow Class Reference . . . 26
2.12.1 Detailed Description . . . 26
2.12.2 Constructor & Destructor Documentation . . . 26
2.12.3 Member Function Documentation . . . 27
2.13 MapController Class Reference . . . 27
2.13.1 Detailed Description . . . 27
2.13.2 Constructor & Destructor Documentation . . . 27
2.13.3 Member Function Documentation . . . 28
2.13.4 Event Documentation . . . 29
2.14 MapWindow Class Reference . . . 29
2.14.1 Detailed Description . . . 30
2.14.2 Constructor & Destructor Documentation . . . 30
2.15 MinuteConverter Class Reference . . . 30
2.15.1 Detailed Description . . . 30
2.15.2 Member Function Documentation . . . 30
2.16 PhoneNumberConverter Class Reference . . . 31
2.16.1 Detailed Description . . . 31
2.16.2 Member Function Documentation . . . 31
2.17 Pinger Class Reference . . . 32

2.17.1 Detailed Description . . . 32
2.17.2 Constructor & Destructor Documentation . . . 32
2.17.3 Member Function Documentation . . . 32
2.18 PriorityConverter Class Reference . . . 33
2.18.1 Detailed Description . . . 33
2.18.2 Member Function Documentation . . . 33
2.19 Settings Class Reference . . . 34
2.19.1 Detailed Description . . . 34
2.19.2 Constructor & Destructor Documentation . . . 34
2.19.3 Member Function Documentation . . . 34
2.19.4 Member Data Documentation . . . 35
2.20 SettingsWindow Class Reference . . . 35
2.20.1 Detailed Description . . . 36
2.20.2 Constructor & Destructor Documentation . . . 36
2.21 SpeexCompression Class Reference . . . 36
2.21.1 Detailed Description . . . 36
2.21.2 Member Function Documentation . . . 36
2.22 StateConverter Class Reference . . . 37
2.22.1 Detailed Description . . . 37
2.22.2 Member Function Documentation . . . 38

1 Namespace Documentation

1.1 Package GraphClass

Classes

- class Graph

  The class adds a graph to the given grid. A user can add points, set bytes per second and scroll to the end of the graph.

1.2 Package Hake_WPF

Namespaces

- package AudioVideoManagers
- package Conversion
- package Network
- package Properties

Classes

- class App

  The Interaction logic for App.xaml.
- class Assignment

The assigment listboxitem contains many properties like handlers, location, time, state, priority, pushpins, assigment info and street name. It sets the content and background depending on the state and the priority and if there is a location available.

- class AsyncReceiver

  The class receives data from the server which invokes an event.
- class Connection

  The class handles the communication between the server and this client. A user must use connect method to connect the server. After that the user can use rest of the methods to manage connection. Also the client should handle most of the events, but at least connectionupdatedevent.
- class MainWindow

  The interaction logic for MainWindow.xaml. A user can ask information from mobile client including video and location. The user can also send tutorials. The user can also manage assigments which are shown in the listbox and in the map by the pushpins.
- class MapController

  The class controls the maps. A User can add a specific pushpin to the maps or add all pushpins to the map again. The user can add maps and also center the maps. It also handles closing the maps.
- class MapWindow

  The class is a simple resizable window only contains a map.
- class Settings

  The class is used to store a object specified with key. A user can add, update, get and remove the objects using the key.
- class SettingsWindow

  The interaction logic for SettingsWindow.xaml. It is just a UI for the settings.

Functions

- delegate void ConnectionsUpdated (ConnectionDto[] newConnections)

  The class delegate is invoked when the connection receives an update.
- delegate void TcpMediaDataReceived (object sender, MediaInformationDto info, byte[] data)

  The class delegate is invoked when TcpMedia is received.
- delegate void UdpMediaDataReceived (object sender, MediaPacket media)

  The class delegate is invoked when udp media is received.
- delegate void MeasurementDataReceivedDelegate (object sender, EventArgs e, MeasurementInstrumentDto instrument, byte[] measurementData, int dataSamplesPerSecond)

  The class delegate is invoked when measurement data media is received.

1.2.1 Function Documentation

1.2.1.1 delegate void Hake_WPF.ConnectionsUpdated (ConnectionDto[] *newConnections*)

The class delegate is invoked when the connection receives an update.

Parameters

newConnections	List of the connections received from the server.

1.2.1.2 delegate void Hake_WPF.MeasurementDataReceivedDelegate (object *sender,* EventArgs *e,* MeasurementInstrumentDto *instrument,* byte[] *measurementData,* int *dataSamplesPerSecond*)

The class delegate is invoked when measurement data media is received.

Parameters

sender	Source object that invokes this delegate.
e	Delegate event arguments.
instrument	Sending instrument.
measurement↩ Data	Recived measurement data.
dataSamples↩ PerSecond	Samples per second for recived measurement data.

1.2.1.3 delegate void Hake_WPF.TcpMediaDataReceived (object *sender,* MediaInformationDto *info,* byte[ ] *data*)

The class delegate is invoked when TcpMedia is received.

Parameters

sender	Source object that invokes this delegate.
info	Information about recived data.
data	Recived data packet.

1.2.1.4 delegate void Hake_WPF.UdpMediaDataReceived (object *sender,* MediaPacket *media*)

The class delegate is invoked when udp media is received.

Parameters

sender	Source object that invokes this delegate.
media	Recived media packet.

1.3 Package Hake_WPF.AudioVideoManagers

Classes

- class AudioVideoTransferManager

  The class is used for processing incoming and outgoing audio and images. It handles media data received from the server. It reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.
- class BufferWaveStream

  This class provides a NAudio WaveStream with an infinite length to facilitate streaming audio playback. It 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 a 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

  The class provides static methods for compression and decompression of speex encoded audio segments.

Functions

- delegate void JpgImageReceived (object sender, byte[] imageData)

  The class delegate for image received events.

1.3.1 Function Documentation

1.3.1.1 delegate void Hake_WPF.AudioVideoManagers.JpgImageReceived (object *sender,* byte[ ] *imageData*)

The class delegate for image received events.

Parameters

sender	The publishing AudioVideoTransferManager instance.
imageData	The byte representation of a jpg compressed image.

1.4 Package Hake_WPF.Conversion

Classes

- class AddressConverter

  The class converts PersonalInfomationDto string that is (Streetname, postalcode, locality). The Converback method is not implemented and returns always null!
- class BoolConverter

  The class converts bool to string. True is 'on' and false is 'ei'.
- class CompressionHelper

  The class for static compression methods. It contains methods to compress and decompress data.
- class LocationConverter

  The class converts the location to string.
- class MinuteConverter

  The class is used for converting the time in minutes to a string and back.
- class PhoneNumberConverter

  The class converter for phonenumbers. It is used to convert a array of phonenumbers to a string.
- class PriorityConverter

  The class converter for priority to a string and back.
- class StateConverter

  The class converts the state of a assignment to a string.

1.5 Package Hake_WPF.Network

Classes

- class Pinger

  The class is used for sending keep-alive messages to the server.

1.6 Package Hake_WPF.Properties

Classes

- class **Resources**

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

2 Class Documentation

2.1 AddressConverter Class Reference

The class converts PersonalInfomationDto string that is (Streetname, postalcode, locality). The Converback method is not implemented and returns always null!

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, CultureInfo culture)

  Tries to cast value to PersonalInformationDto and then returns a string that is combination of streetname, postalcode and locality. Does not show postalcode if it is set to -1.
- object ConvertBack (object value, Type targetType, object parameter, CultureInfo culture)

  Method to convert string back to PersonalInformationDto. Returns always null because this method is not implemented.

2.1.1 Detailed Description

The class converts PersonalInfomationDto string that is (Streetname, postalcode, locality). The Converback method is not implemented and returns always null!

<author>Atte Söderlund</author>

2.1.2 Member Function Documentation

2.1.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

Tries to cast value to PersonalInformationDto and then returns a string that is combination of streetname, postalcode and locality. Does not show postalcode if it is set to -1.

Parameters

value	Given value that should be type of PersonalInformationDto.
targetType	This is ignored in the code so do not use this.
parameter	Parameter is not in use so do not use this.
culture	Culture is not in use so do not use this.

Returns

Address as string.

2.1.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

Method to convert string back to PersonalInformationDto. Returns always null because this method is not implemented.

Parameters

value	String.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns the orginal value.

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

- AddressConverter.cs

2.2 App Class Reference

The Interaction logic for App.xaml.

Inherits Application.

2.2.1 Detailed Description

The Interaction logic for App.xaml.

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

- App.xaml.cs

2.3 Assignment Class Reference

The assigment listboxitem contains many properties like handlers, location, time, state, priority, pushpins, assigment info and street name. It sets the content and background depending on the state and the priority and if there is a location available.

Inherits ListBoxItem, and INotifyPropertyChanged.

Public Types

- enum priorities { **NotUrgent** = 0, **Urgent** = 1 }

  Not urgent and urgent enums. NotUrgent is 0 and Urgent is 1 as integers.
- enum States {
  New = 0, **InProgress** = 1, **InTransfer** = 2, **Finalized** = 3,
  Hold = 4 }

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

Public Member Functions

- Assignment (String guid, DateTime time, priorities priority, States state, LocationInformationDto location)

  The function initializes Assigment by setting state, priority, time, location, assimentinfo, background color and content.
- Assignment (String guid, DateTime time, priorities priority, States state)

  The constructor without location.

Static Public Attributes

- static readonly DependencyProperty IsHandlerProperty

  DependencyProperty for ishandler. It is used to determine currently active assignment.

Properties

- bool **ClientConnected** `[get, set]`
- String EmergencyType `[get, set]`

  Sets and gets the emergency type. Emergency type can be any string that is short description about emergency.
- ObservableCollection
  < TextMessageDto > TextMessages `[get, set]`

  Textmessages in observablecollection. It is a collection of received and sent textmessages.
- PersonalInformationDto PersonalInfo `[get, set]`

  Sets and gets the personal information as PersonalInformationDto that includes name, address and list of phone numbers all as strings.
- priorities Priority `[get, set]`

  Sets and gets priority of the assignment. When the value is set, it also sets background and the content.
- String Guid `[get, set]`

  Sets and gets the guid. It is used to identify the connection.
- DateTime Time `[get, set]`

Sets and gets time. The time specifies when the assignment is taken in. When the time is setted, it also sets the background color and the content.

- bool NoSound `[get, set]`

  Gets and sets NoSound. It specifies whether caller can speak or not.

- States State `[get, set]`

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

- bool IsHandler `[get, set]`

  Sets and gets IsHandler. It is used to determine currently active assignment.

- LocationInformationDto Location `[get, set]`

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

- double LocationAccuracyMeters `[get, set]`

  Sets and gets the locationAccuracyMeters. It specifies how accurate the received GPS location is.

- DateTimeOffset LocationAcquisitionTime `[get, set]`

  Sets and gets LocationAcquisitionTime. It specifies the time when the location information is taken.

- ObservableCollection< Pushpin > Pushpins `[get, set]`

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

- String StreetName `[get, set]`

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

- MobileDeviceInformationDto DeviceInfo `[get, set]`

  Sets and gets the DeviceInfo. It contains information about the callers device.

Events

- PropertyChangedEventHandler PropertyChanged

  Is used to notify that a property has changed.

2.3.1 Detailed Description

The assigment listboxitem contains many properties like handlers, location, time, state, priority, pushpins, assigment info and street name. It sets the content and background depending on the state and the priority and if there is a location available.

<author>Atte Söderlund</author>

2.3.2 Member Enumeration Documentation

2.3.2.1 enum priorities

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

2.3.2.2 enum States

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

2.3.3 Constructor & Destructor Documentation

2.3.3.1 Assignment (String *guid,* DateTime *time,* priorities *priority,* States *state,* LocationInformationDto *location*)

The function initializes Assigment by setting state, priority, time, location, assimentinfo, background color and content.

Parameters

time	The assigment connection time
priority	The assigments priority from Assigment.Priorities
state	The assigments state from Assigment.States
location	The assigments location

2.3.3.2 Assignment (String *guid,* DateTime *time,* priorities *priority,* States *state*)

The constructor without location.

Parameters

time	The assigment connection time
priority	The assigments priority from Assigment.Priorities
state	The assigments state from Assigment.States

2.3.4 Member Data Documentation

2.3.4.1 readonly DependencyProperty IsHandlerProperty [static]

Initial value:

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

DependencyProperty for ishandler. It is used to determine currently active assignment.

2.3.5 Property Documentation

2.3.5.1 MobileDeviceInformationDto DeviceInfo [get], [set]

Sets and gets the DeviceInfo. It contains information about the callers device.

2.3.5.2 String EmergencyType [get], [set]

Sets and gets the emergency type. Emergency type can be any string that is short description about emergency.

2.3.5.3 String Guid [get], [set]

Sets and gets the guid. It is used to identify the connection.

2.3.5.4 bool IsHandler [get], [set]

Sets and gets IsHandler. It is used to determine currently active assignment.

2.3.5.5 LocationInformationDto Location [get], [set]

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

2.3.5.6 double LocationAccuracyMeters [get], [set]

Sets and gets the locationAccuracyMeters. It specifies how accurate the received GPS location is.

2.3.5.7 DateTimeOffset LocationAcquisitionTime [get], [set]

Sets and gets LocationAcquisitionTime. It specifies the time when the location information is taken.

2.3.5.8 bool NoSound [get], [set]

Gets and sets NoSound. It specifies whether caller can speak or not.

2.3.5.9 PersonalInformationDto PersonalInfo [get], [set]

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

2.3.5.10 priorities Priority [get], [set]

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

2.3.5.11 ObservableCollection<Pushpin> Pushpins [get], [set]

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

2.3.5.12 States State [get], [set]

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

2.3.5.13 String StreetName [get], [set]

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

2.3.5.14 ObservableCollection<TextMessageDto> TextMessages [get], [set]

Textmessages in observablecollection. It is a collection of received and sent textmessages.

2.3.5.15 DateTime Time [get], [set]

Sets and gets time. The time specifies when the assignment is taken in. When the time is setted, it also sets the background color and the content.

2.3.6 Event Documentation

2.3.6.1 PropertyChangedEventHandler PropertyChanged

Is used to notify that a property has changed.

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

- Assignment.cs

2.4 AsyncReceiver Class Reference

The class receives data from the server which invokes an event.

Inherits IWcfCallCenterServiceCallback.

Public Member Functions

- void ActiveConnectionsUpdated (ConnectionDto[] updatedConnections)

  The class delegate is invoked everytime the client receives an update for assignment from the server. This method only invokes ConnectionsUpdatedEvent so Connection class can handle it.
- void AudioVideoReceived (string sourceGuid, MediaInformationDto mediaInfo, byte[] mediaData)

  The function invokes tcpMediaDataReceivedEvent.

- void MeasurementDataReceived (string sourceGuid, MeasurementInstrumentDto instrument, byte[] measurementData)

  The function invokes MeasurementDateReceivedEvent.

Public Attributes

- TcpMediaDataReceived TcpMediaDataReceivedEvent

  It is a delegate that is triggered when new media is received.
- MeasurementDataReceivedDelegate MeasurementDataReceivedEvent

  It is a delegate that is triggered when meausrement data is received.

Events

- ConnectionsUpdated ConnectionsUpdatedEvent

  It is a event that is triggered when the server tells the client that the connections have changed.

2.4.1 Detailed Description

The class receives data from the server which invokes an event.

2.4.2 Member Function Documentation

2.4.2.1 void ActiveConnectionsUpdated (ConnectionDto[] *updatedConnections*)

The class delegate is invoked everytime the client receives an update for assignment from the server. This method only invokes ConnectionsUpdatedEvent so Connection class can handle it.

<author>Veli-Mikko Puupponen</author>

Parameters

updated↩ Connections	List of changed connections.

2.4.2.2 void AudioVideoReceived (string *sourceGuid,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function invokes tcpMediaDataReceivedEvent.

2.4.2.3 void MeasurementDataReceived (string *sourceGuid,* MeasurementInstrumentDto *instrument,* byte[] *measurementData*)

The function invokes MeasurementDateReceivedEvent.

2.4.3 Member Data Documentation

2.4.3.1 MeasurementDataReceivedDelegate MeasurementDataReceivedEvent

It is a delegate that is triggered when meausrement data is received.

2.4.3.2 TcpMediaDataReceived TcpMediaDataReceivedEvent

It is a delegate that is triggered when new media is received.

2.4.4 Event Documentation

2.4.4.1 ConnectionsUpdated ConnectionsUpdatedEvent

It is a event that is triggered when the server tells the client that the connections have changed.

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

- Connection.cs

2.5 AudioVideoTransferManager Class Reference

The class is used for processing incoming and outgoing audio and images. It handles media data received from the server. It reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.

Public Member Functions

- AudioVideoTransferManager (Connection c)

  The function initializes a new AudioVideoTransferManager that uses the provided connection to receive and transmit audio and pictures.
- void EnableOutgoingAudio ()

  The function enables audio recording and publishing to the server from the primary audio capture in the system.
- void DisableOutgoingAudio ()

  The function stops recording audio and publishing it to the server.
- void EnableIncomingAudio ()

  The function starts receiving audio from the server and reproducing it using the primary audio output device in the system.
- void DisableIncomingAudio ()

  The function stops receiving and reproducing audio.

Public Attributes

- JpgImageReceived JpgImageReceivedEvent

  It is a event handler for JpgImageReceivedEvent delegate.

Properties

- bool UseOutgoingUdp `[get, set]`

  Gets and sets the UseOutGoingUdp. It specifies whether to use UDP or not.

2.5.1 Detailed Description

The class is used for processing incoming and outgoing audio and images. It handles media data received from the server. It reproduces speex encoded audio and Wave file segments using the primary audio output device in the system.

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

It publishes a JpgImageReceivedEvent upon receiving an image from the server.

2.5.2 Constructor & Destructor Documentation

2.5.2.1 AudioVideoTransferManager (Connection *c*)

The function initializes a new AudioVideoTransferManager that uses the provided connection to receive and transmit audio and pictures.

Parameters

c	The valid Connection instance.

2.5.3 Member Function Documentation

2.5.3.1 void DisableIncomingAudio ( )

The function stops receiving and reproducing audio.

2.5.3.2 void DisableOutgoingAudio ( )

The function stops recording audio and publishing it to the server.

2.5.3.3 void EnableIncomingAudio ( )

The function starts receiving audio from the server and reproducing it using the primary audio output device in the system.

2.5.3.4 void EnableOutgoingAudio ( )

The function enables audio recording and publishing to the server from the primary audio capture in the system.

2.5.4 Member Data Documentation

2.5.4.1 JpgImageReceived JpgImageReceivedEvent

It is a event handler for JpgImageReceivedEvent delegate.

2.5.5 Property Documentation

2.5.5.1 bool UseOutgoingUdp [get],[set]

Gets and sets the UseOutGoingUdp. It specifies whether to use UDP or not.

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

- AudioVideoTransferManager.cs

2.6 BoolConverter Class Reference

The class converts bool to string. True is 'on' and false is 'ei'.

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, CultureInfo culture)
  The function converts bool to string.
- object ConvertBack (object value, Type targetType, object parameter, CultureInfo culture)
  The function converts String to bool.

2.6.1 Detailed Description

The class converts bool to string. True is 'on' and false is 'ei'.

<author>Atte Söderlund</author>

2.6.2 Member Function Documentation

2.6.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts bool to string.

Parameters

value	Bool that needs converting to string.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Bool as String.

2.6.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts String to bool.

Parameters

value	"true" or "false" strings.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Bool whether can parse it from string or not.

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

- BoolConverter.cs

2.7 BufferWaveStream Class Reference

This class provides a NAudio WaveStream with an infinite length to facilitate streaming audio playback. It 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 a pcm audio available in the buffer, it will be returned to the reader, possibly padded with silence to meet the desired read length.

Inherits WaveStream.

Public Member Functions

- BufferWaveStream (int sampleRate, int bytesPerSample, int channels)

  The function 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)

  The function writes the provided audio data to the outgoing PCM segment buffer.

- override int Read (byte[] buffer, int offset, int count)

  The function reads PCM samples from the underlying buffer. If no data is available, returns silent simples.

Properties

- override WaveFormat WaveFormat `[get]`

  Gets the streams format type.
- override long Length `[get]`

  Gets the length of stream.
- override long Position `[get, set]`

  Sets and gets the position. It specifies the position of stream.

2.7.1 Detailed Description

This class provides a NAudio WaveStream with an infinite length to facilitate streaming audio playback. It 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 a 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>

2.7.2 Constructor & Destructor Documentation

2.7.2.1 BufferWaveStream (int *sampleRate,* int *bytesPerSample,* int *channels*)

The function initializes a new BufferWaveStream that has a WaveFormat defined by the provided parameters. The audio buffer is empty and has no length limit.

Parameters

sampleRate	The aample rate im samples per second.
bytesPerSample	The number of bytes per PCM sample.
channels	The number of channels.

2.7.3 Member Function Documentation

2.7.3.1 override int Read (byte[] *buffer,* int *offset,* int *count*)

The function reads PCM samples from the underlying buffer. If no data is available, returns silent simples.

Parameters

buffer	The bffer to which the data is copied to.
offset	The offset for the data at the target buffer.
count	The desired count of data.

Returns

Processed data.

2.7.3.2 override void Write (byte[] *buffer,* int *offset,* int *count*)

The function writes the provided audio data to the outgoing PCM segment buffer.

Parameters

buffer	The bytes to write.

offset	The offset at which the bytes to write start.
count	The count of the bytes to write<param>

2.7.4 Property Documentation

2.7.4.1 override long Length [get]

Gets the length of stream.

2.7.4.2 override long Position [get], [set]

Sets and gets the position. It specifies the position of stream.

2.7.4.3 override WaveFormat WaveFormat [get]

Gets the streams format type.

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

- BufferWaveStream.cs

2.8 CompressionHelper Class Reference

The class for static compression methods. It contains methods to compress and decompress data.

Static Public Member Functions

- static byte[] CompressGZip (byte[] sourceData, bool useOptimalCompression)

  The function compresses the provided byte array using the GZip algorithm provided by System.IO.Compression library.
- static byte[] DecompressGZip (byte[] sourceData)

  The function decompresses the provided byte array using the GZip algorithm provided by System.IO.Compression library.

2.8.1 Detailed Description

The class for static compression methods. It contains methods to compress and decompress data.

<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.

2.8.2 Member Function Documentation

2.8.2.1 static byte [] CompressGZip (byte[] *sourceData,* bool *useOptimalCompression*) [static]

The function compresses the provided byte array using the GZip algorithm provided by System.IO.Compression library.

Parameters

sourceData	The data to be compressed.
useOptimal↩ Compression	True, if using optimal compression method, otherwise using fastest.

Returns

The input data compressed with GZip.

2.8.2.2 static byte [] DecompressGZip (byte[] *sourceData*) [static]

The function decompresses the provided byte array using the GZip algorithm provided by System.IO.Compression library.

Parameters

sourceData	The data to be decompressed.

Returns

The input data decompressed with GZip.

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

- CompressionHelper.cs

2.9 Connection Class Reference

The class handles the communication between the server and this client. A user must use connect method to connect the server. After that the user can use rest of the methods to manage connection. Also the client should handle most of the events, but at least connectionupdatedevent.

Public Member Functions

- delegate void AssigmentUpdated (ObservableCollection< Assignment > newConnections)

  The class delegate is invoked when the connection is updated.
- delegate void AssignmentTakenForHandling ()

  The class delegate is invoked when the current connection is setted.
- Connection (String username, String password)

  The function creates a connection with the credentials provided.
- ObservableCollection< Assignment > Connect (String endpointAddress)

  The function creates a receiver and connects to the server with wcf and udp.
- bool UdpSendMedia (MediaInformation mediaInfo, byte[] data, int originalLength)

  The function sends media to the server using the active UdpMediaClientSocket instance. It returns true, if the data is successfully queued for sending. Otherwise returns false.
- void ChangePriority (Assignment assignment, Assignment.priorities priority)

  The function chances the priority to current connection and then updates that to the connections list and finaly to the server. It makes the connection in a new thread so it won't block the ui thread.
- ObservableCollection< Assignment > ConvertConnectionsToAssigments (ConnectionDto[] Connections)

  The function converts ConnectionDto to Assignment. If the location is not set, it uses null and do not assign the accuracymeters or the time from the location. If device info is empty, uses empty DeviceInfoDto.
- void RequestAction (RemoteActionDto action)

  The function requests an action for the current user connection from caller with the given action. CurrentConnection must be specified or this function does nothing.
- void RequestMeasurementData (MeasurementInstrumentDto instrument)

  If there is current connection specified, this function requests the measurements data to start from the server with the given instrument.
- void CancelMeasurementData (MeasurementInstrumentDto instrument)

  The function cancels the request to the server to stop measurement data sending. The Current connection is needed for this.

- bool IsConnected ()

  The function returns boolean whether the user is connected to server or not.
- ObservableCollection< Assignment > GetAllConnections ()

  The function requests all connections from the server and converts the server response to the assignment list. This function should be used only one time to save localy and then use ActiveConnectionsUpdated to update those.
- bool Reconnect ()

  The function reconnects to the server.
- bool Disconnect ()

  The function disconnects from the server.
- void ProcessAssignment (object assignment)

  The function changes the assignments state to in progress to the server. First it search the right connection using guid and then makes the connection as current connection.
- void SendTextMessage (String message)

  The function makes a new thread to send a message to server so it won't block ui thread.
- void RequestAudioVideoMedia ()

  The function makes a new thread to request audio video so it won't block the ui thread.
- void RequestAudioMedia ()

  The function makes a new thread to request audio so it won't block the ui thread.
- void CancelAudioVideoMedia ()

  The function makes a new thread to request cancel of the audio and video so it won't block the ui thread.
- void CancelOnlyPicture ()

  The function makes a new thread to request cancel of the picture so it won't block the ui thread.
- async Task PingAsync (int interval)

  The function sends a keepalive message to the server

Public Attributes

- UdpMediaDataReceived UdpMediaDataReceivedEvent

  The class delegate is invoked when udp media is received.

Properties

- AsyncReceiver receiver `[get, set]`

  The function gets and sets AsyncReceiver. It handles the connection between the server and the client.

Events

- AssigmentUpdated AssigmentUpdatedEvent

  The event is triggered when the assinment is updated.
- AssignmentTakenForHandling AssignmentTakenForHandlingEvent

  The event is triggered when the assinment is taken to handlin.

2.9.1 Detailed Description

The class handles the communication between the server and this client. A user must use connect method to connect the server. After that the user can use rest of the methods to manage connection. Also the client should handle most of the events, but at least connectionupdatedevent.

<author>Atte Söderlund</author>

2.9.2 Constructor & Destructor Documentation

2.9.2.1 Connection (String *username,* String *password*)

The function creates a connection with the credentials provided.

Parameters

username	The username that connection uses.
password	The password that connection uses.

2.9.3 Member Function Documentation

2.9.3.1 delegate void AssigmentUpdated (ObservableCollection< **Assignment** > *newConnections*)

The class delegate is invoked when the connection is updated.

Parameters

newConnections	The connections as Assignments.

2.9.3.2 delegate void AssignmentTakenForHandling ( )

The class delegate is invoked when the current connection is setted.

2.9.3.3 void CancelAudioVideoMedia ( )

The function makes a new thread to request cancel of the audio and video so it won't block the ui thread.

2.9.3.4 void CancelMeasurementData (MeasurementInstrumentDto *instrument*)

The function cancels the request to the server to stop measurement data sending. The Current connection is needed for this.

<author>Niko Mononen</author>

Parameters

instrument	The instrument that is sending the data.

2.9.3.5 void CancelOnlyPicture ( )

The function makes a new thread to request cancel of the picture so it won't block the ui thread.

2.9.3.6 void ChangePriority (**Assignment** *assignment,* **Assignment.priorities** *priority*)

The function chances the priority to current connection and then updates that to the connections list and finaly to the server. It makes the connection in a new thread so it won't block the ui thread.

Parameters

assignment	The assignment that needs updating.
priority	The priority for the assignment to be changed.

2.9.3.7 ObservableCollection<**Assignment**> Connect (String *endpointAddress*)

The function creates a receiver and connects to the server with wcf and udp.

Returns

ObservableCollection of connections as assignments.

2.9.3.8 ObservableCollection<**Assignment**> ConvertConnectionsToAssigments (ConnectionDto[] *Connections*)

The function converts ConnectionDto to Assignment. If the location is not set, it uses null and do not assign the accuracymeters or the time from the location. If device info is empty, uses empty DeviceInfoDto.

Parameters

Connection	ConnectionDto that needs converting to Assigment.

Returns

Converted Assigment.

2.9.3.9 bool Disconnect ( )

The function disconnects from the server.

Returns

True when disconnection is carried out without an error and false if there is any error.

2.9.3.10 ObservableCollection<Assignment> GetAllConnections ( )

The function requests all connections from the server and converts the server response to the assignment list. This function should be used only one time to save localy and then use ActiveConnectionsUpdated to update those.

Returns

ObservableCollection of all connections as Assignment.

2.9.3.11 bool IsConnected ( )

The function returns boolean whether the user is connected to server or not.

Returns

Boolean whether connection established or not.

2.9.3.12 async Task PingAsync (int *interval*)

The function sends a keepalive message to the server

<author>Ilkka Rautiainen</author>

Parameters

interval	The ping interval in seconds.

Returns

task that sends keepalive messages to the server.

2.9.3.13 void ProcessAssignment (object *assignment*)

The function changes the assignments state to in progress to the server. First it search the right connection using guid and then makes the connection as current connection.

Parameters

assignment	The assignment that is to be set as in progress.

2.9.3.14 bool Reconnect ( )

The function reconnects to the server.

Returns

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

2.9.3.15 void RequestAction (RemoteActionDto *action*)

The function requests an action for the current user connection from caller with the given action. CurrentConnection must be specified or this function does nothing.

Parameters

action	What action should be carried out.

2.9.3.16 void RequestAudioMedia ( )

The function makes a new thread to request audio so it won't block the ui thread.

2.9.3.17 void RequestAudioVideoMedia ( )

The function makes a new thread to request audio video so it won't block the ui thread.

2.9.3.18 void RequestMeasurementData (MeasurementInstrumentDto *instrument*)

If there is current connection specified, this function requests the measurements data to start from the server with the given instrument.

<author>Niko Mononen</author>

Parameters

instrument	The instrument that provides data.

2.9.3.19 void SendTextMessage (String *message*)

The function makes a new thread to send a message to server so it won't block ui thread.

Parameters

message	Message to be send.

2.9.3.20 bool UdpSendMedia (MediaInformation *mediaInfo,* byte[ ] *data,* int *originalLength*)

The function sends media to the server using the active UdpMediaClientSocket instance. It returns true, if the data is successfully queued for sending. Otherwise returns false.

Parameters

mediaInfo	The description of the media.
data	The bytes constituting the media data.
originalLength	The length of the media prior to the encoding for such encodings that require the length for decompression.

Returns

True, if the data is queued for sending, otherwise false.

2.9.4 Member Data Documentation

2.9.4.1 UdpMediaDataReceived UdpMediaDataReceivedEvent

The class delegate is invoked when udp media is received.

2.9.5 Property Documentation

2.9.5.1 AsyncReceiver receiver [get],[set]

The function gets and sets AsyncReceiver. It handles the connection between the server and the client.

2.9.6 Event Documentation

2.9.6.1 AssigmentUpdated AssigmentUpdatedEvent

The event is triggered when the assinment is updated.

2.9.6.2 AssignmentTakenForHandling AssignmentTakenForHandlingEvent

The event is triggered when the assinment is taken to handlin.

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

- Connection.cs

2.10 Graph Class Reference

The class adds a graph to the given grid. A user can add points, set bytes per second and scroll to the end of the graph.

Public Member Functions

- Graph (Grid grid)

  The function makes canvas to the given grid with a black background and initializes a green line that connetcts the points that are added by addpoint method.
- void SetBytesPerSecond (int bps)

  The function sets the Bytes per second for the data.
- void AddPoint (int y)

  The function adds a point to the graph and scales the points so that scrvollviewers width is 1 second of measurement data.
- void ScrollToEnd ()

  The function scrolls to the end of the graph.

2.10.1 Detailed Description

The class adds a graph to the given grid. A user can add points, set bytes per second and scroll to the end of the graph.

<author>Niko Mononen</author>

2.10.2 Constructor & Destructor Documentation

2.10.2.1 Graph (Grid *grid*)

The function makes canvas to the given grid with a black background and initializes a green line that connetcts the points that are added by addpoint method.

Parameters

grid	The Grid where this graph is added.

2.10.3 Member Function Documentation

2.10.3.1 void AddPoint (int *y*)

The function adds a point to the graph and scales the points so that scrvollviewers width is 1 second of measurement data.

Parameters

y	The y point of the graph.

2.10.3.2 void ScrollToEnd ( )

The function scrolls to the end of the graph.

2.10.3.3 void SetBytesPerSecond (int *bps*)

The function sets the Bytes per second for the data.

Parameters

bps	The value in bytes per second.

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

- GraphClass.cs

2.11 LocationConverter Class Reference

The class converts the location to string.

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)
  The function casts value to LocationInformation type and then turns it to a string with 4 decimals.
- object ConvertBack (object value, Type targetType, object parameter, System.Globalization.CultureInfo culture)

2.11.1 Detailed Description

The class converts the location to string.

<author>Atte Söderlund</author>

2.11.2 Member Function Documentation

2.11.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* System.Globalization.CultureInfo *culture*)

The function casts value to LocationInformation type and then turns it to a string with 4 decimals.

Parameters

value	The LocationInformationDto to be converted to string.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

The Location as string with 4 decimals on longitude and latitude.

2.11.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* System.Globalization.CultureInfo *culture*)

Converts string back to Location.

Parameters

value	A string.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns the prority converted from the value.

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

- LocationConverter.cs

2.12 MainWindow Class Reference

The interaction logic for MainWindow.xaml. A user can ask information from mobile client including video and location. The user can also send tutorials. The user can also manage assigments which are shown in the listbox and in the map by the pushpins.

Inherits Window.

Public Member Functions

- MainWindow ()

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

Protected Member Functions

- override void OnClosing (System.ComponentModel.CancelEventArgs e)

  The function disconnects from the server before closing program. Also if the mapwindows or/and the settingswindow is open, it closes those too.

2.12.1 Detailed Description

The interaction logic for MainWindow.xaml. A user can ask information from mobile client including video and location. The user can also send tutorials. The user can also manage assigments which are shown in the listbox and in the map by the pushpins.

<author>Atte Söderlund</author>

2.12.2 Constructor & Destructor Documentation

2.12.2.1 MainWindow ( )

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

2.12.3 Member Function Documentation

2.12.3.1 override void OnClosing (System.ComponentModel.CancelEventArgs *e*) [protected]

The function disconnects from the server before closing program. Also if the mapwindows or/and the settingswindow is open, it closes those too.

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

- MainWindow.xaml.cs

2.13 MapController Class Reference

The class controls the maps. A User can add a specific pushpin to the maps or add all pushpins to the map again. The user can add maps and also center the maps. It also handles closing the maps.

Public Member Functions

- delegate void MapWindowClosing ()

  Invoked when MapWindow is closing. It is used to tell that map window is closed.
- MapController ()

  The function is constructor. It does nothing.
- void AddMap (Map map)

  The function adds the given map to the maps collection.
- void MapToOwnWindow (ObservableCollection< Assignment > assignments)

  The function makes a new map window and adds that map to the maps collections. Then it adds pushpins to the maps and adds event handler for the map window closing.
- void Close ()

  The function closes the map window if it is open.
- void AddAllPushpins (ObservableCollection< Assignment > assignments)

  The function adds puhspins to all maps.
- void CenterMaps (Assignment assignment)

  The function centers all maps to the assignments location.

Events

- MapWindowClosing MapWindowClosingEvent

  This event is sent when MapWindow is closing.

2.13.1 Detailed Description

The class controls the maps. A User can add a specific pushpin to the maps or add all pushpins to the map again. The user can add maps and also center the maps. It also handles closing the maps.

<author>Atte Söderlund</author>

2.13.2 Constructor & Destructor Documentation

2.13.2.1 MapController ( )

The function is constructor. It does nothing.

2.13.3 Member Function Documentation

2.13.3.1 void AddAllPushpins (ObservableCollection< Assignment > *assignments*)

The function adds puhspins to all maps.

Parameters

assignments	The list of assignmets that contains pushpins.

2.13.3.2 void AddMap (Map *map*)

The function adds the given map to the maps collection.

Parameters

map	The given map.

2.13.3.3 void CenterMaps (Assignment *assignment*)

The function centers all maps to the assignments location.

Parameters

assignment	The assignment that contains location.

2.13.3.4 void Close ( )

The function closes the map window if it is open.

2.13.3.5 void MapToOwnWindow (ObservableCollection< Assignment > *assignments*)

The function makes a new map window and adds that map to the maps collections. Then it adds pushpins to the maps and adds event handler for the map window closing.

Parameters

assignments	The list of assignments for puhspins to be added.

2.13.3.6 delegate void MapWindowClosing ( )

Invoked when MapWindow is closing. It is used to tell that map window is closed.

2.13.4 Event Documentation

2.13.4.1 MapWindowClosing MapWindowClosingEvent

This event is sent when MapWindow is closing.

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

- MapController.cs

2.14 MapWindow Class Reference

The class is a simple resizable window only contains a map.

Inherits Window.

Public Member Functions

- MapWindow (Location location, int zoomLevel)

  The function initializes MapWindow centered to the given location and the given zoom level.

2.14.1 Detailed Description

The class is a simple resizable window only contains a map.

2.14.2 Constructor & Destructor Documentation

2.14.2.1 MapWindow (Location *location,* int *zoomLevel*)

The function initializes MapWindow centered to the given location and the given zoom level.

Parameters

location	The location where to center the map.
zoomLevel	The wanted zoom level.

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

- MapWindow.xaml.cs

2.15 MinuteConverter Class Reference

The class is used for converting the time in minutes to a string and back.

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, CultureInfo culture)
  The function converts the minutes to a string of format "x h x min".
- object ConvertBack (object value, Type targetType, object parameter, CultureInfo culture)
  The function converts a string of format "x h x min" to minutes.

2.15.1 Detailed Description

The class is used for converting the time in minutes to a string and back.

<author>Ilkka Rautiainen</author>

2.15.2 Member Function Documentation

2.15.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts the minutes to a string of format "x h x min".

Parameters

value	The object that contains the integer value to convert.
targetType	The type that defines the target format for conversion.
parameter	The optional parameters.
culture	The culture information.

Returns

A string of format "x h x min".

2.15.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts a string of format "x h x min" to minutes.

Parameters

value	The object that contains the string value to convert.
targetType	The type that defines the target format for conversion.
parameter	The optional parameters.
culture	The culture information.

Returns

The total converted minutes.

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

- MinuteConverter.cs

2.16 PhoneNumberConverter Class Reference

The class converter for phonenumbers. It is used to convert a array of phonenumbers to a string.

Inherits IValueConverter.

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)
  Method to convert phone numbers back to original. Returns always null because this method is not implemented.

2.16.1 Detailed Description

The class converter for phonenumbers. It is used to convert a array of phonenumbers to a string.

<author>Atte Söderlund</author>

2.16.2 Member Function Documentation

2.16.2.1 object Convert (object *value*, Type *targetType*, object *parameter*, CultureInfo *culture*)

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

Parameters

value	The string[] of phone numbers.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns phone numbers as string.

2.16.2.2 object ConvertBack (object *value*, Type *targetType*, object *parameter*, CultureInfo *culture*)

Method to convert phone numbers back to original. Returns always null because this method is not implemented.

Parameters

value	String.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns the orginal value.

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

- PhoneNumberConverter.cs

2.17 Pinger Class Reference

The class is used for sending keep-alive messages to the server.

Public Member Functions

- Pinger (int interval, Connection connection)

  The function creates the pinger.
- void StartPinger ()

  The function starts the pinger by creating a new timer and an event handler. The keep-alive message interval is entered here.

2.17.1 Detailed Description

The class is used for sending keep-alive messages to the server.

<author>Ilkka Rautiainen</author>

2.17.2 Constructor & Destructor Documentation

2.17.2.1 Pinger (int *interval,* Connection *connection*)

The function creates the pinger.

Parameters

interval	The interval of keep-alive messages in seconds.
connection	The connection.

2.17.3 Member Function Documentation

2.17.3.1 void StartPinger ( )

The function starts the pinger by creating a new timer and an event handler. The keep-alive message interval is entered here.

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

- Pinger.cs

2.18 PriorityConverter Class Reference

The class converter for priority to a string and back.

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, CultureInfo culture)

  The function converts a priority to a string.
- object ConvertBack (object value, Type targetType, object parameter, CultureInfo culture)

  The function converts a string back to the priority.

2.18.1 Detailed Description

The class converter for priority to a string and back.

<author>Atte Söderlund</author>

2.18.2 Member Function Documentation

2.18.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts a priority to a string.

Parameters

value	The priority.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in conversion.
culture	Not in use.

Returns

Return the priority as a string

2.18.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts a string back to the priority.

Parameters

value	The string.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns prority converted from the value.

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

- PriorityConverter.cs

2.19 Settings Class Reference

The class is used to store a object specified with key. A user can add, update, get and remove the objects using the key.

Public Member Functions

- Settings ()

  The function initializes settings by reading them from settings.cfg.
- void AddOrUpdate (String key, object obj)

  The function adds or updates a object with the given key.
- void Remove (String key)

  The function removes a object with the given key.
- object Get (String key)

  The function returns object from settings that is stored using given key.
- bool Save ()

  The function saves the settings to the file settings.cfg if its not in use.

Public Attributes

- const String ENDPOINTADDRESS = "endpointAddress"

  The key for a object that contains the endpoint address.
- const String NEWURGENTSTATECOLOR = "newUrgentStateColor"

  The key for a object that contains the color for an assignment that is new and urgent.
- const String GRAPHLINECOLOR = "graphLineColor"

  The key for a object that contains the color of the graph line.
- const String GRAPHBACKGROUNDCOLOR = "graphBackgroundColor"

  The key for a object that contains the background color of the graph.

2.19.1 Detailed Description

The class is used to store a object specified with key. A user can add, update, get and remove the objects using the key.

<author>Atte Söderlund</author>

2.19.2 Constructor & Destructor Documentation

2.19.2.1 Settings ( )

The function initializes settings by reading them from settings.cfg.

2.19.3 Member Function Documentation

2.19.3.1 void AddOrUpdate (String *key*, object *obj*)

The function adds or updates a object with the given key.

Parameters

key	The key that specifies the object.
obj	The object to store.

2.19.3.2 object Get (String *key*)

The function returns object from settings that is stored using given key.

Parameters

key	Key string.

Returns

The stored object corresponding to the key.

2.19.3.3 void Remove (String *key*)

The function removes a object with the given key.

Parameters

key	The key that is used to remove a object.

2.19.3.4 bool Save ( )

The function saves the settings to the file settings.cfg if its not in use.

Returns

True if no exceptions catched and false otherwise.

2.19.4 Member Data Documentation

2.19.4.1 const String ENDPOINTADDRESS = "endpointAddress"

The key for a object that contains the endpoint address.

2.19.4.2 const String GRAPHBACKGROUNDCOLOR = "graphBackgroundColor"

The key for a object that contains the background color of the graph.

2.19.4.3 const String GRAPHLINECOLOR = "graphLineColor"

The key for a object that contains the color of the graph line.

2.19.4.4 const String NEWURGENTSTATECOLOR = "newUrgentStateColor"

The key for a object that contains the color for an assignment that is new and urgent.

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

- Settings.cs

2.20 SettingsWindow Class Reference

The interaction logic for SettingsWindow.xaml. It is just a UI for the settings.

Inherits Window.

Public Member Functions

- SettingsWindow ()

  THe function initializes settings by loading an example data.

2.20.1 Detailed Description

The interaction logic for SettingsWindow.xaml. It is just a UI for the settings.

<author>Atte Söderlund</author>

2.20.2 Constructor & Destructor Documentation

2.20.2.1 SettingsWindow ( )

THe function initializes settings by loading an example data.

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

- SettingsWindow.xaml.cs

2.21 SpeexCompression Class Reference

The class provides static methods for compression and decompression of speex encoded audio segments.

Static Public Member Functions

- static byte[] DecompressSpeex (SpeexDecoder decoder, byte[] data, int originatingLength)

  The function decompresses the provided MediaPackets of payload data using the provided SpeexDecoder.It returns the resulting PCM samples in a byte array.
- static int CompressSpeex (byte[] pcmSegment, int pcmByteCount, int sampleSizeBytes, SpeexEncoder encoder, out byte[] compressed)

  The function encodes the provided PCM samples using the provided SpeexEncoder. The resulting encoded bytes are put into the compressed array. It returns the count of 16 bit samples that was used as the input of the speex encoder.

2.21.1 Detailed Description

The class provides static methods for compression and decompression of speex encoded audio segments.

<author>Veli-Mikko Puupponen</author>

2.21.2 Member Function Documentation

2.21.2.1 static int CompressSpeex (byte[] *pcmSegment,* int *pcmByteCount,* int *sampleSizeBytes,* SpeexEncoder *encoder,* out byte[] *compressed*) [static]

The function encodes the provided PCM samples using the provided SpeexEncoder. The resulting encoded bytes are put into the compressed array. It 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 of 320, the differece is padded with silence after the samples.

The SpeexEncoder is assumed to be operating in the BandMode.Wide and sampleSizeBytes is assumed to be 2, i.e. 16bit PCM.

Parameters

pcmSegment	16Bit PCM samples to be encode into speex
sampleSize↩ Bytes	The size of the PCM samples in bytes, it should be 2.
encoder	SpeexEncoder used to encode the data
compressed	The target array for the resulting speex encoded audio.

Returns

The count of 16Bit PCM samples compressed.

2.21.2.2 static byte [ ] DecompressSpeex (SpeexDecoder *decoder,* byte[ ] *data,* int *originatingLength*) [static]

The function decompresses the provided MediaPackets of payload data using the provided SpeexDecoder.It returns the 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, it returns an empty array.

Parameters

decoder	SpeexDecodes instance used to decode the speex encoding.
data	Speex encoded audio.
originating↩ Length	The count of 16 bit PCM samples that were encoded to produce the provided encoded data.

Returns

The resulting PCM samples or an empty array.

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

- SpeexCompression.cs

2.22 StateConverter Class Reference

The class converts the state of a assignment to a string.

Inherits IValueConverter.

Public Member Functions

- object Convert (object value, Type targetType, object parameter, CultureInfo culture)

  The function converts a state to a string by searching it from states array and using its the index to cast it to a state.
- object ConvertBack (object value, Type targetType, object parameter, CultureInfo culture)

  Method to convert state back to original. Returns always null because this method is not implemented.

2.22.1 Detailed Description

The class converts the state of a assignment to a string.

<author>Atte Söderlund</author>

2.22.2 Member Function Documentation

2.22.2.1 object Convert (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

The function converts a state to a string by searching it from states array and using its the index to cast it to a state.

Parameters

value	A state.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in conversion.

Returns

The state as string.

2.22.2.2 object ConvertBack (object *value,* Type *targetType,* object *parameter,* CultureInfo *culture*)

Method to convert state back to original. Returns always null because this method is not implemented.

Parameters

value	A string.
targetType	Not in use. Specifies target type.
parameter	Not in use. Converting parameters.
culture	Not in use. Culture info to be used in the conversion.

Returns

Returns the orginal state.

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

- StateConverter.cs