Halyri - Server

1.0

Generated by Doxygen 1.8.7

Sat Jun 21 2014 20:15:56

Contents

1 Namespace Documentation

1.1 Package HalyriServer

Namespaces

- package Controllers
- package Model
- package Services
- package Udp

Classes

- class AutomapperInitialization

  The class contains a static method for setting up AutoMapper mappings for the application.
- class Global

  The class is generated code. Server uses it. Do not modify.
- class Startup

  This class is mostly generated code to start the application. It also starts SignalR.

1.2 Package HalyriServer.Controllers

Classes

- class CallCenterController

  The singleton class manages call center client connections. It contains a dictionary mapping active CallCenter↩ Connections to their identifying GUIDs. All call center client requests from and to the corresponding WCF interface WcfCallCenterService are passed through the class.
- class ConnectionController

  The singleton class manages active emergency connections (Connection). Contains a dictionary for active connections and their identifying GUIDs. All emergency connection state and life cycle methods are handled by this class. It functions as a broker between the mobile emergency connection client and the call center client.
- class DatabaseController

  The class is for database interaction. It handles database connections and operations for storing and retrieving Connection-instances for persistency. It uses EntityFramework.
- class DataTransferController

  The class is for managing audio/video and measurement data distribution from the mobile emergency clients to the associated call center clients. The data packets from the mobile emergency clients can be relayed in a fan-out style to multiple call center connections. The data subscriptions for each call center connection are stored in the "data↩ Subscribers" dictionary.
- class MobileClientController

  The class is for interacts with an emergency client. it functions as a broker between the ConnectionController and the SignalR and WCF interfaces used by the mobile emergency client. Maintains a reference to the SignalR IHub↩ ConnectionContext required for the communication from the server to the connected mobile emergency clients.

1.3 Package HalyriServer.Model

Namespaces

- package Business
- package Transfer

1.4 Package HalyriServer.Model.Business

Classes

- class ConnectionDatabaseContext

  The class defines database for the server.
- class TextMessage

  The class is used for text based short messages. It stores the message content, the message originator and the server timestamp of the message.

Enumerations

- enum ConnectionPriority { NotUrgent, Urgent }

  The enumeration represents the urgency level of an emergency connection in the system.
- enum ConnectionState {
  Arrived, InProgress, InTransfer, Processed,
  Hold }

  The emergency connection The enumeration identifies the state of an emergency connection in the system.

1.4.1 Enumeration Type Documentation

1.4.1.1 enum ConnectionPriority

The enumeration represents the urgency level of an emergency connection in the system.

<author>Veli-Mikko Puupponen, Ilkka Rautiainen</author>

Enumerator

NotUrgent The not urgent connections have the normal priority.

Urgent The urgent connections have the high priority.

1.4.1.2 enum ConnectionState

The emergency connection The enumeration identifies the state of an emergency connection in the system.

<author>Veli-Mikko Puupponen</author>

Enumerator

Arrived The connection that has arrived from the mobile emergency client to the system and is waiting to be handled by a call center connection.

InProgress The connection has one or more call center connections currently handling it.

InTransfer The connection is being transferred from one handling call center connection to other handling call center connection(s)

Processed The connection is already handled and is no longer active.

Hold The connections in hold have been opened for handling but the handling call center connection has put them on hold or has unexpectedly disconnected, leaving the connection in hold state.

1.5 Package HalyriServer.Model.Transfer

Classes

- class AudioVideoContainerDto

  The dto container class stores a MediaInformationDto instance, 32bit int sequence number, data length prior to the specified compression, a boolean value indicating whether the instance has any payload and the payload byte array.

- class CallCenterConnectionDto

  The dto corresponds to the CallCenterConnection class.
- class ConnectionDto

  The dto container class corresponds to the Connection class.
- class ConnectionLatencyInformationDto

  The dto container class corresponds to the ConnectionLatencyInformation class.
- class EmergencyTypeDto

  The dto class corresponds to the EmergencyType class.
- class LocationInformationDto

  The dto class coresponds to the LocationInformation class.
- class MeasurementInstrumentDto

  The dto class corresponds to the MeasurementInstrument class.
- class MediaConfigurationDto

  The dto container class represents a audio/video capture configuration. It contains quantifiers specifying the quality of the captured media.
- class MediaInformationDto

  The dto container class for audio/video media description. It contains a property identifying if the payload data has been compressed with GZip for transport. Contains a MediaTypeDto instance describing the media format.
- class MedicalInformationDto

  The dto class is for medical information.
- class MobileDeviceInformationDto

  The dto class corresponds to the MobileDeviceInformation class.
- class PersonalInformationDto

  The dto class corresponds to the PersonalInformation class.
- class TextMessageDto

  The dto class corresnpods to the TextMessage class.
- class UserCredentialsDto

  The dto container class is used for user login credentials. It contains a user name and a password.

Enumerations

- enum ConnectionPriorityDto { NotUrgent, Urgent }

  The dto corresponds to the ConnectionPriority The enumeration. It represents the urgency level of an emergency connection in the system.
- enum ConnectionStateDto {
  Arrived, InProgress, InTransfer, Processed,
  Hold }

  The dto corresponding to the ConnectionState The enumeration. It identifies the state of an emergency connection in the system.
- enum LocationTypeDto { Initial, Response, Movement, UserSpecified }

  The dto class corresponds to the LocationType The enumeration.
- enum MeasurementInstrumentTypeDto { ECG }

  The dto class corresnpods to the MeasurementInstrumentType The enumeration.
- enum MediaTypeDto {
  Wave, Jpeg, AAC, H264,
  MP4, Opus, Speex }

  The enumeration represents some common audio and video/picture encoding and container formats.
- enum RemoteActionDto { requestLocation, requestDeviceStatusinformation, requestShowLocationMap, requestInstrumentList }

  The enumeration identifies parameterless operations executed by the remote mobile emergency client.

1.5.1 Enumeration Type Documentation

1.5.1.1 enum ConnectionPriorityDto

The dto corresponds to the ConnectionPriority The enumeration. It represents the urgency level of an emergency connection in the system.

<author>Veli-Mikko Puupponen</author>

Enumerator

NotUrgent NotUrgent connections have a normal priority.

Urgent Urgent connections have a high priority.

1.5.1.2 enum ConnectionStateDto

The dto corresponding to the ConnectionState The enumeration. It identifies the state of an emergency connection in the system.

<author>Veli-Mikko Puupponen</author>

Enumerator

Arrived The connection that has arrived from the mobile emergency client to the system and is waiting to be handled by a call center connection.

InProgress The connection has one or more call center connections currently hanling it.

InTransfer The connection is being transferred from one handling call center connection to other handling call center connection(s)

Processed The connection is already handled and is no longer active.

Hold The connections in hold have been opened for handling but the handling call center connection has put them on hold or has unexpectedly disconnected, leaving the connection in hold state.

1.5.1.3 enum LocationTypeDto

The dto class corresponds to the LocationType The enumeration.

<author>Veli-Mikko Puupponen</author> The enumeration for location information type. Differentiates manually entered location and a location derived from GPS data. Also differentiates GPS coordinates collected initially and as a resault of client movement exceeding the set threshold level.

Enumerator

Initial The location information that is gathered by a background process or automatically at the beginning of the connection.

Response The location information is sent as a response to a request originating from the server.

Movement The location information is automatically sent as client movement exceeds the set treshold level.

UserSpecified The location information was manually entered by a user.

1.5.1.4 enum MeasurementInstrumentTypeDto

The dto class corresnpods to the MeasurementInstrumentType The enumeration.

<author>Veli-Mikko Puupponen</author> It identifies different measurement instrument types.

Enumerator

ECG Represents electrocardiography instrument.

1.5.1.5 enum MediaTypeDto

The enumeration represents some common audio and video/picture encoding and container formats.

<author>Veli-Mikko Puupponen</author>

Enumerator

Wave The enumeration member representing the Wave file. format.

Jpeg The enumeration member representing the Jpg. compression.

AAC The enumeration member representing the AAC. audio encoding.

H264 The enumeration member representing the h264. video encoding.

MP4 The enumeration member representing the MP4. file container.

Opus The enumeration member representing Opus. audio encoding.

Speex The enumeration member representing Speex. audio encoding.

1.5.1.6 enum RemoteActionDto

The enumeration identifies parameterless operations executed by the remote mobile emergency client.

Enumerator

requestLocation The enumeration member representing a GPS location update request.

requestDeviceStatusinformation The enumeration member representing a device status information update request.

requestShowLocationMap The enumeration member representing a request to show a location map to the emergency client user.

requestInstrumentList The enumeration member representing a request for an updated list of the measurement instruments attached to the mobile device.

1.6 Package HalyriServer.Services

Namespaces

- package Exceptions

Classes

- interface IMobileClientMethods

  The interface defines all callback methods supported by the mobile emergency client's SignalR client.
- interface IWcfCallCenterCallback

  The interface defines the callback methods for the call center clients.
- interface IWcfCallCenterService

  The interface defines the WCF service for the call center clients.
- interface IWcfMobileService

  The interface defines the WCF service for the emergency mobile clients. The interface is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.
- class SignalRMobileHub

  SignalR hub for the mobile emergency client. It supports emergency client connect, disconnect and reconnect methods. Functions as the callback channel for server-to-client asynchronous method invocations.
- class WcfCallCenterService

  The WCF service for call center clients. It contains function interface for call center clients. It implements IWcfCall↩ CenterService.

See also

HalyriServer.Services.IWcfCallCenterService

- class WcfMobileService

  The WCF service for the emergency mobile clients. The service is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.

1.7 Package HalyriServer.Services.Exceptions

Classes

- class ConnectionFault

  The fault class is used for exceptions related to connection state.
- class Fault

  The base class is used for all fault exceptions transferred through the WCF interfaces of the server.
- class ParameterFault

  The fault class is used for exceptions related to method parameters.
- class TargetStateFault

  The fault class is used for exceptions related to target instance state exceptions.

1.8 Package HalyriServer.Udp

Classes

- class UdpMediaRelayServerCore

  The media broker service core is used with the UdpMediaClientSocket instances. It implements ping reply and simple routing mechanism. Accepts routing configuration packets (ControlPacket). For every guid, there can be a single target guid to which all MediaHeaderPackets and MediaContinuationPackets from that guid are sent. If there is no valid target mapping for the guid specified in the received media packet, the packet is silently ignored.

Functions

- delegate void StatusMessage (string msg)

  The class delegate for loggable status message events.

1.8.1 Function Documentation

1.8.1.1 delegate void HalyriServer.Udp.StatusMessage (string *msg*)

The class delegate for loggable status message events.

Parameters

msg	The status message string.

2 Class Documentation

2.1 AudioVideoContainerDto Class Reference

The dto container class stores a MediaInformationDto instance, 32bit int sequence number, data length prior to the specified compression, a boolean value indicating whether the instance has any payload and the payload byte array.

Public Member Functions

- AudioVideoContainerDto ()

  The function initializes a new AudioVideoContainerDto that has not payload data or media type description. If not modified later, the instance will have its IsInitializedContainer set to false.
- AudioVideoContainerDto (MediaInformationDto info, byte[] data)

  The function initializes a new AudioVideoContainerDto containing the provided payload media data and the provided MediaInformationDto instance describing the data. the instance will have IsInitializedContainer set to true.

Properties

- MediaInformationDto MediaInfo `[get, set]`

  The function gets or sets the MediaInformationDto describing the compression format and the media encoding of the payload data.
- byte[] MediaData `[get, set]`

  The function gets or sets the paylod data byte array.
- int SequenceNumber `[get, set]`

  The function gets or sets the sequence number of this media packet.
- int OriginatingDataLength `[get, set]`

  The function gets or sets the length of the data prior to applying the encoding described by the MediInfo.
- bool IsInitializedContainer `[get, set]`

  The function gets or sets value indicating whether the instance is initialized and contains payload data. True, if there is payload, otherwise false.

2.1.1 Detailed Description

The dto container class stores a MediaInformationDto instance, 32bit int sequence number, data length prior to the specified compression, a boolean value indicating whether the instance has any payload and the payload byte array.

<author>Veli-Mikko Puupponen</author> It is used for transfer of audio/video media over the WCF services.

2.1.2 Constructor & Destructor Documentation

2.1.2.1 AudioVideoContainerDto ( )

The function initializes a new AudioVideoContainerDto that has not payload data or media type description. If not modified later, the instance will have its IsInitializedContainer set to false.

2.1.2.2 AudioVideoContainerDto (MediaInformationDto *info*, byte[] *data*)

The function initializes a new AudioVideoContainerDto containing the provided payload media data and the provided MediaInformationDto instance describing the data. the instance will have IsInitializedContainer set to true.

Parameters

info	The MediaInformationDto specifying payload data properties.
data	The payload media data.

2.1.3 Property Documentation

2.1.3.1 bool IsInitializedContainer `[get], [set]`

The function gets or sets value indicating whether the instance is initialized and contains payload data. True, if there is payload, otherwise false.

2.1.3.2 byte [] MediaData [get], [set]

The function gets or sets the paylod data byte array.

2.1.3.3 MediaInformationDto MediaInfo [get], [set]

The function gets or sets the MediaInformationDto describing the compression format and the media encoding of the payload data.

2.1.3.4 int OriginatingDataLength [get], [set]

The function gets or sets the length of the data prior to applying the encoding described by the MediInfo.

2.1.3.5 int SequenceNumber [get], [set]

The function gets or sets the sequence number of this media packet.

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

- AudioVideoContainerDto.cs

2.2 AutomapperInitialization Class Reference

The class contains a static method for setting up AutoMapper mappings for the application.

Static Public Member Functions

- static void initializeAutomapperMappings ()

  The function The function initializes all auto mapper mappings between server entity classes and data transfer classes used over WCF and SignalR interfaces.

2.2.1 Detailed Description

The class contains a static method for setting up AutoMapper mappings for the application.

<author>Veli-Mikko Puupponen</author>

2.2.2 Member Function Documentation

2.2.2.1 static void initializeAutomapperMappings ( ) [static]

The function The function initializes all auto mapper mappings between server entity classes and data transfer classes used over WCF and SignalR interfaces.

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

- AutomapperInitialization.cs

2.3 CallCenterConnection Class Reference

The class is for call center client connection. Contains a reference to the WCF callback channel for the connected client and the identifying GUID for the connected client. Contains the callback methods for publishing data to the connected client.

Public Member Functions

- CallCenterConnection (CallCenterController callCenterController)

  The function initializes a new CallCenterConnection instance that reports changes its connection state to the provided CallCenterController.
- void PushUpdatedConnections (List< ConnectionDto > updatedConnections)

  The function sends a list of ConnectionDtos to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.
- void PushAudioVideo (string sourceGuid, MediaInformationDto mediaInfo, byte[] mediaData)

  The function sends audio/video received from emergency mobile client to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.
- void PushMeasurementData (string sourceGuid, MeasurementInstrumentDto instrument, byte[] measurement↩ Data)

  The function sends measurement data from emergency mobile client to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.

Properties

- string Id `[get, set]`

  The function gets or sets the primary key identifying this call center connection in the database.
- string ConnectionGuid `[get, set]`

  The function gets or sets the Guid used to identify the instance in the system.
- string UserName `[get, set]`

  The function gets or sets the identifying user name for this client.
- string Password `[get, set]`

  The function gets or sets the password for this client,
- IWcfCallCenterCallback ClientCallbackChannel `[get, set]`

  The function gets or sets the IWcfCallCenterCallback instance used for client callbacks.
- Boolean IsInactive `[get, set]`

  True, if a callback attempt for this connection has time out or the client has explicitly disconnected. New callbacks will not be performed on a timed out connection. False for all active connections.

2.3.1 Detailed Description

The class is for call center client connection. Contains a reference to the WCF callback channel for the connected client and the identifying GUID for the connected client. Contains the callback methods for publishing data to the connected client.

<author>Veli-Mikko Puupponen</author>

2.3.2 Constructor & Destructor Documentation

2.3.2.1 CallCenterConnection (CallCenterController *callCenterController*)

The function initializes a new CallCenterConnection instance that reports changes its connection state to the provided CallCenterController.

Parameters

callCenterController	The callCenterController instance to which the instance reports changes in its connection state.

2.3.3 Member Function Documentation

2.3.3.1 void PushAudioVideo (string *sourceGuid,* **MediaInformationDto** *mediaInfo,* byte[] *mediaData*)

The function sends audio/video received from emergency mobile client to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.

Parameters

sourceGuid	The guid of the originating mobile emergency client connection.
mediaInfo	The description of the media.
mediaData	The media data.

2.3.3.2 void PushMeasurementData (string *sourceGuid,* **MeasurementInstrumentDto** *instrument,* byte[] *measurementData*)

The function sends measurement data from emergency mobile client to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.

Parameters

sourceGuid	The guid of the originating mobile emergency client connection.
instrument	The instrument producing the data.
measurementData	The measurement data.

2.3.3.3 void PushUpdatedConnections (List< **ConnectionDto** > *updatedConnections*)

The function sends a list of ConnectionDtos to the call center client represented by the instance. If the operation times out, the instance is marked as HasTimeOut = true, and subsequent calls to this method will not try to do a callback to the client before HasTimeOut has been set to false again.

Parameters

updatedConnections	The list of updated ConnectionDtos.

2.3.4 Property Documentation

2.3.4.1 IWcfCallCenterCallback ClientCallbackChannel [get],[set]

The function gets or sets the IWcfCallCenterCallback instance used for client callbacks.

2.3.4.2 string ConnectionGuid [get],[set]

The function gets or sets the Guid used to identify the instance in the system.

2.3.4.3 string Id [get],[set]

The function gets or sets the primary key identifying this call center connection in the database.

2.3.4.4 Boolean IsInactive [get], [set]

True, if a callback attempt for this connection has time out or the client has explicitly disconnected. New callbacks will not be performed on a timed out connection. False for all active connections.

2.3.4.5 string Password [get], [set]

The function gets or sets the password for this client,

2.3.4.6 string UserName [get], [set]

The function gets or sets the identifying user name for this client.

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

- CallCenterConnection.cs

2.4 CallCenterConnectionDto Class Reference

The dto corresponds to the CallCenterConnection class.

Properties

- string ConnectionGuid [get, set]

  The function gets or sets the Guid used for identifying the instance in the system.
- string UserName [get, set]

  The function gets or sets the user name identifying this client.

2.4.1 Detailed Description

The dto corresponds to the CallCenterConnection class.

<author>Veli-Mikko Puupponen</author> It contains only the Guid used to identify the call center client in the system and the userName used to identify the client to other clients in the client programs.

2.4.2 Property Documentation

2.4.2.1 string ConnectionGuid [get], [set]

The function gets or sets the Guid used for identifying the instance in the system.

2.4.2.2 string UserName [get], [set]

The function gets or sets the user name identifying this client.

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

- CallCenterConnectionDto.cs

2.5 CallCenterController Class Reference

The singleton class manages call center client connections. It contains a dictionary mapping active CallCenter↩ Connections to their identifying GUIDs. All call center client requests from and to the corresponding WCF interface WcfCallCenterService are passed through the class.

Public Member Functions

- CallCenterConnectionDto CallCenterClientConnects (UserCredentialsDto clientCredentials, IWcfCall↩ CenterCallback clientCallbackChannel)

  The method for opening a new active call center connection. User credentials supplied in the UserCredentialsDto is currently copied to the instantialized CallCenterConnection and no checking is performed.
- List< ConnectionDto > ClientGetActiveConnections (CallCenterConnectionDto client)

  The function gets a list of all active emergency connections currently known to ConnectionController
- void PublishUpdatedActiveConnections (List< ConnectionDto > updatedConnections)

  The function sends the emergency connections provided to all active call center connections. If CallCenterConnection has HasTimedOut = true, it is considered inactive.
- void CallCenterClientReconnects (CallCenterConnectionDto user, IWcfCallCenterCallback callback)

  The method handles call center client reconnections. Callback channel associated with the client is updated and the connection is set to not have timed out.
- void CallCenterClientDisconnects (CallCenterConnectionDto user)

  The method handlies call center client disconnect. The disconnected clients are no longer marked as associated handlers in their open emergency connections.
- void CallCenterClientConnectionFaulted (CallCenterConnection user)

  The function handles call center connection fault evenets. It sets the connection status to inactive and requests the connection controller to remove the call center client from the associated handlers from any emergency connections.
- void ClientOpenConnectionForProcessing (CallCenterConnectionDto user, ConnectionDto emergency↩ Connection)

  The function adds the call center client as a handler to the emergency connection.
- void ClientTransferOpenConnection (CallCenterConnectionDto user, List< CallCenterConnectionDto > targetCallCenterConnections)

  The function starts transfer of an open emergency connection to the provided call center connection(s). The requesting call center client must be an active handler of the connection to be transferred.
- void ClientSetConnectionPriority (CallCenterConnectionDto user, ConnectionDto connection)

  The function changes priority on the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.
- void ClientRequestPutOpenConnectionOnHold (CallCenterConnectionDto user, ConnectionDto connection)

  The function puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.
- void ClientRequestMarkProcessedCloseOpenConnection (CallCenterConnectionDto user, ConnectionDto connection)

  The function closes the emergency connection and marks it processed. If there are multiple attached handlers to the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.
- void ClientRequestRemoteAction (CallCenterConnectionDto user, ConnectionDto connection, Remote↩ ActionDto action)

  The function requests an action to be performed by the provided remote mobile emergency client
- void ClientRequestMediaUpstreaming (CallCenterConnectionDto user, ConnectionDto connection, Media↩ ConfigurationDto mediaConfiguration)

  The function requests the mobile emergency client to start upstreaming media according to the provided configuration.
- void ClientRequestMediaDownstreaming (CallCenterConnectionDto user, ConnectionDto connection, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.
- void ClientRequestStartMeasurement (CallCenterConnectionDto user, ConnectionDto connection, MeasurementInstrumentDto measurementDevice)

  The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.
- void ClientSendTextMessage (CallCenterConnectionDto user, ConnectionDto connection, TextMessageDto textMessage)

  The function sends a text based message to the specified mobile emergency client.

- void ClientRequestStopMeasurement (CallCenterConnectionDto user, ConnectionDto connection, MeasurementInstrumentDto measurementDevice)

  The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.
- bool ClientUploadedMediaSegment (CallCenterConnectionDto user, MediaInformationDto mediaInfo, byte[] mediaData)

Properties

- static CallCenterController Instance `[get]`

  The function returns the CallCenterController instance.

2.5.1 Detailed Description

The singleton class manages call center client connections. It contains a dictionary mapping active CallCenterConnections to their identifying GUIDs. All call center client requests from and to the corresponding WCF interface WcfCallCenterService are passed through the class.

<author>Veli-Mikko Puupponen, Ilkka Rautiainen</author> New connections are added to the dictionary on a successful CallCenterClientConnects method call. This singleton is instantialized by the first call to the WcfCallCenterService or by ConnectionController.

See also

HalyriServer.Services.IWcfCallCenterService, HalyriServer.Services.WcfCallCenterService, HalyriServer.Controllers.ConnectionController

2.5.2 Member Function Documentation

2.5.2.1 void CallCenterClientConnectionFaulted (CallCenterConnection *user*)

The function handles call center connection fault evenets. It sets the connection status to inactive and requests the connection controller to remove the call center client from the associated handlers from any emergency connections.

Parameters

user	The CallCenterConnection representing an existing call center connection.

2.5.2.2 CallCenterConnectionDto CallCenterClientConnects (UserCredentialsDto *clientCredentials*, IWcfCallCenterCallback *clientCallbackChannel*)

The method for opening a new active call center connection. User credentials supplied in the UserCredentialsDto is currently copied to the instantialized CallCenterConnection and no checking is performed.

A guid generated by the ConnectionController is assigned to the new connection and the connection is mapped in active collection dictionary with it.

Parameters

clientCredentials	The UserCredentialsDto containing credentials identifying the connecting user.
clientCallbackChannel	The IWcfCallCenterCallback associated with the client connection.

Returns

The CallCenterConnectionDto representing the call center connection.

2.5.2.3 void CallCenterClientDisconnects (CallCenterConnectionDto *user*)

The method handlies call center client disconnect. The disconnected clients are no longer marked as associated handlers in their open emergency connections.

It Throws ConnectionFault if the provided CallCenterConnection is not a valid active connection.

Parameters

user	The CallCenterConnectionDto for the disconnecting user.

2.5.2.4 void CallCenterClientReconnects (CallCenterConnectionDto *user,* IWcfCallCenterCallback *callback*)

The method handles call center client reconnections. Callback channel associated with the client is updated and the connection is set to not have timed out.

It Throws ConnectionFault if the provided CallCenterConnection is not a valid active connection.

Parameters

user	The CallCenterConnectionDto for the reconnecting user.
callback	The IWcfCallCenterCallback for the connection.

2.5.2.5 List<ConnectionDto> ClientGetActiveConnections (CallCenterConnectionDto *client*)

The function gets a list of all active emergency connections currently known to ConnectionController

It Throws ConnectionFault if the provided CallCenterConnection is not a valid active connection.

Parameters

client	The client requesting call center connection.

Returns

A list of active emergency connections.

2.5.2.6 void ClientOpenConnectionForProcessing (CallCenterConnectionDto *user,* ConnectionDto *emergencyConnection*)

The function adds the call center client as a handler to the emergency connection.

It Throws ConnectionFault if the provided call center connection does not exist in the system.

Parameters

user	The CallCenterConnectionDto of the call center connection.
emergency↩ Connection	The ConnectionDto of the emergency connection.

2.5.2.7 void ClientRequestMarkProcessedCloseOpenConnection (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function closes the emergency connection and marks it processed. If there are multiple attached handlers to the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto of the requester.
connection	The ConnectionDto of the target emergency connection.

2.5.2.8 void ClientRequestMediaDownstreaming (CallCenterConnectionDto *user,* ConnectionDto *connection,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
mediaUrl	The location of the media to be played back at the mobile client.

2.5.2.9 void ClientRequestMediaUpstreaming (CallCenterConnectionDto *user,* ConnectionDto *connection,* MediaConfigurationDto *mediaConfiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
media↩ Configuration	The quality parameters for the requested media.

2.5.2.10 void ClientRequestPutOpenConnectionOnHold (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.

It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto of the requester.
connection	The ConnectionDto of the target emergency connection.

2.5.2.11 void ClientRequestRemoteAction (CallCenterConnectionDto *user,* ConnectionDto *connection,* RemoteActionDto *action*)

The function requests an action to be performed by the provided remote mobile emergency client

It Throws FaultException<ConnectionFault> if such connection does not currently exists in active connections collection. It Throws FaultException<TargetStateFault> if the connection is already processed, has disconnected or the requester is not an associated handler for this connection.

Parameters

user	The CallCenterConnectionDto of the requester.
connection	The ConnectionDto of the target emergency connection.
action	The cation to be performed by the remote mobile emergency client.

2.5.2.12 void ClientRequestStartMeasurement (CallCenterConnectionDto *user,* ConnectionDto *connection,* MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device To be enabled and started uploading data from.

2.5.2.13 void ClientRequestStopMeasurement (CallCenterConnectionDto *user,* ConnectionDto *connection,* MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device to disable and stop uploading data from.

2.5.2.14 void ClientSendTextMessage (CallCenterConnectionDto *user,* ConnectionDto *connection,* TextMessageDto *textMessage*)

The function sends a text based message to the specified mobile emergency client.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
textMessage	The message to be sent.

2.5.2.15 void ClientSetConnectionPriority (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function changes priority on the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

It Throws FaultException<ConnectionFault> if the call center connection does not exist. It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto of the requester.
connection	The ConnectionDto of the target emergency connection.

2.5.2.16 void ClientTransferOpenConnection (CallCenterConnectionDto *user,* List< CallCenterConnectionDto > *targetCallCenterConnections*)

The function starts transfer of an open emergency connection to the provided call center connection(s). The requesting call center client must be an active handler of the connection to be transferred.

It Throws TargetStateFault if the connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto of the transferring connection.
targetCall↩ Center↩ Connections	The callCenterConnectionDtos of the target connections.

TODO

2.5.2.17 bool ClientUploadedMediaSegment (CallCenterConnectionDto *user,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function Publish media from the given user to the target client specified in the media informaton. queue. This data can be asynchronously read by the remote client over its WCF connection.

Parameters

user	The CallCenterConnectionDto describing the publishing call center client.
mediaInfo	The MediaInformationDto describing the published media data.
mediaData	The media data.

Returns

Boolean whether the publishing succseeded or not.

2.5.2.18 void PublishUpdatedActiveConnections (List< ConnectionDto > *updatedConnections*)

The function sends the emergency connections provided to all active call center connections. If CallCenter↩ Connection has HasTimedOut = true, it is considered inactive.

Parameters

updated↩ Connections	The list of ConnectionDto instances to be sent to all active call center connections.

2.5.3 Property Documentation

2.5.3.1 CallCenterController Instance [static],[get]

The function returns the CallCenterController instance.

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

- CallCenterController.cs

2.6 Connection Class Reference

Class representing a mobile emergency client connection. Contains the GUID identifying the client, the SignalR connection ID for asynchronous callbacks, time stamp of the connection arrival, current state of the connection and

all information received from the client or sent to it during its connection.

Public Member Functions

- Connection ()

  The function instantiates a Connection that is not connected to a remote mobile emergency client.
- Connection (MobileClientController controller, string guid, string transportId)

  The function instantiates a Connection that is connected to a remote mobile emergency client using the provided MobileClientController and the SignalR transportId. Connected instances will automatically propagate changes of certain attributes to the mobile emergy client.

Properties

- int Id `[get, set]`

  The function gets or sets the primary key identifying this connection in the database.
- string ConnectionGuid `[get, set]`

  The function gets or sets the Guid idenfying this connection in the system.
- string TransportId `[get, set]`

  The function gets or sets the SignalR transport ID identifying this connection on the SignalR Hub instance.
- bool RequestedNoSound `[get, set]`

  The function gets or sets the silent operation request state. True, if the mobile emergency client represented by the instance has requested operation without audio.
- PersonalInformation PersonalInformation `[get, set]`

  The function gets or sets the personal information provided by the mobile emergency client represented by this instace.
- ConnectionPriority ConnectionPriority `[get, set]`

  The function gets or sets the priority of this emerngecy connection.
- EmergencyType EmergencyType `[get, set]`

  The function gets or sets the type of emergency provided by the mobile emergency client represented by the instance.
- bool AttachedToRemoteClient `[get]`

  The function gets the value indicating whether the instance is attached to an active mobile emergency client or restored from presistent database storage. True, if there is an attached client, otherwise false.
- DateTime ArrivalTime `[get, set]`

  The function gets or sets the DateTime specifying the moment in time when this connection first arrived to the server from the mobile emergency client.
- bool clientDisconnected `[get, set]`

  The function gets or sets the value indicating if the mobile emergency client repserensted by the instance has already disconnected its server connection. True, if the client has disconnected, otherwise false.
- ConnectionState ConnectionState `[get, set]`

  The function gets or sets the state of this connection. If represents an active connection, changes to this property will be propagated to the mobile emergency client associated with this connection unless it has disconnected.
- virtual ICollection
  < MobileDeviceInformation > MobileDeviceInformation `[get]`

  The function gets the collection containing all mobile device status information updates provided by the mobile emergency client during the emergency connection.
- virtual ICollection
  < LocationInformation > LocationInformation `[get]`

  The function gets the collection containing all location information updates provided by the mobile emergency client during the emergency connection.
- virtual ICollection
  < CallCenterConnection > AttachedCallCenterConnections `[get]`

  The function gets the collection containing all CallCenterConnection instances representing the call center clients that are currently handling this emergency connection.

- virtual ICollection
  < MeasurementInstrument > MeasurementInstruments [get, set]

  The function gets the collection containing all MeasurementInstrumentDto instances provided on the mobile emergency client's last available measurement instrument list update.
- virtual ICollection< TextMessage > TextMessages [get]

  The function gets the collection containing all text based messages received from or sent to the mobile emergency client during this emergency connection.

2.6.1 Detailed Description

Class representing a mobile emergency client connection. Contains the GUID identifying the client, the SignalR connection ID for asynchronous callbacks, time stamp of the connection arrival, current state of the connection and all information received from the client or sent to it during its connection.

<author>Veli-Mikko Puupponen</author>

2.6.2 Constructor & Destructor Documentation

2.6.2.1 Connection ( )

The function instantiates a Connection that is not connected to a remote mobile emergency client.

2.6.2.2 Connection (MobileClientController *controller,* string *guid,* string *transportId*)

The function instantiates a Connection that is connected to a remote mobile emergency client using the provided MobileClientController and the SignalR transportId. Connected instances will automatically propagate changes of certain attributes to the mobile emergy client.

Parameters

controller	The MobileClientController for communication to the mobile emergency client.
guid	The guid for this mobile emergency client connection.
transportId	The SignalR transport ID for communication to the mobile emergency client.

2.6.3 Property Documentation

2.6.3.1 DateTime ArrivalTime [get], [set]

The function gets or sets the DateTime specifying the moment in time when this connection first arrived to the server from the mobile emergency client.

2.6.3.2 virtual ICollection<CallCenterConnection> AttachedCallCenterConnections [get]

The function gets the collection containing all CallCenterConnection instances representing the call center clients that are currently handling this emergency connection.

2.6.3.3 bool AttachedToRemoteClient [get]

The function gets the value indicating whether the instance is attached to an active mobile emergency client or restored from presistent database storage. True, if there is an attached client, otherwise false.

2.6.3.4 bool clientDisconnected [get], [set]

The function gets or sets the value indicating if the mobile emergency client repserensted by the instance has already disconnected its server connection. True, if the client has disconnected, otherwise false.

2.6.3.5 string ConnectionGuid [get], [set]

The function gets or sets the Guid idenfying this connection in the system.

2.6.3.6 ConnectionPriority ConnectionPriority [get],[set]

The function gets or sets the priority of this emerngecy connection.

2.6.3.7 ConnectionState ConnectionState [get],[set]

The function gets or sets the state of this connection. If represents an active connection, changes to this property will be propagated to the mobile emergency client associated with this connection unless it has disconnected.

2.6.3.8 EmergencyType EmergencyType [get],[set]

The function gets or sets the type of emergency provided by the mobile emergency client represented by the instance.

2.6.3.9 int Id [get],[set]

The function gets or sets the primary key identifying this connection in the database.

2.6.3.10 virtual ICollection<LocationInformation> LocationInformation [get]

The function gets the collection containing all location information updates provided by the mobile emergency client during the emergency connection.

2.6.3.11 virtual ICollection<MeasurementInstrument> MeasurementInstruments [get],[set]

The function gets the collection containing all MeasurementInstrumentDto instances provided on the mobile emergency client's last available measurement instrument list update.

2.6.3.12 virtual ICollection<MobileDeviceInformation> MobileDeviceInformation [get]

The function gets the collection containing all mobile device status information updates provided by the mobile emergency client during the emergency connection.

2.6.3.13 PersonalInformation PersonalInformation [get],[set]

The function gets or sets the personal information provided by the mobile emergency client represented by this instace.

2.6.3.14 bool RequestedNoSound [get],[set]

The function gets or sets the silent operation request state. True, if the mobile emergency client represented by the instance has requested operation without audio.

2.6.3.15 virtual ICollection<TextMessage> TextMessages [get]

The function gets the collection containing all text based messages received from or sent to the mobile emergency client during this emergency connection.

2.6.3.16 string TransportId [get],[set]

The function gets or sets the SignalR transport ID identifying this connection on the SignalR Hub instance.

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

- Connection.cs

2.7 ConnectionController Class Reference

The singleton class manages active emergency connections (Connection). Contains a dictionary for active connections and their identifying GUIDs. All emergency connection state and life cycle methods are handled by this class. It functions as a broker between the mobile emergency connection client and the call center client.

Public Member Functions

- string GetNewGuidStrig ()

  The function generates a new GUID. It currently uses .NET framework's Guid.NewGuid(). No collision checking is performed. All-zero GUIDs are not returned.

- string MobileClientConnected (string transportId)

  The function handles connection from a new mobile emergency connection client. The client is given a GUID and marked to have arrived at System.DateTime.Now. The instantiated connection is set to be connected to a remote mobile emergency client. Priority of a new connection is always urgent.

- void MobileClientReconnected (string guid, string transportId)

  The function handles reconnection of a mobile emergency connection client. The SignalR connection id for this emergency connection is updated in the corresponding Connection instance.

- void MobileClientDisconnected (string guid)

  The function handles disconnect event of a mobile emegency connection client. The Connection is marked as disconnected unless it already has the status "Processed". All active call center connections are notified about the connection state change through callCenterController.PublishUpdatedActiveConnections if the connection was not already in processed state.

- void MobileClientUpdatedLocation (string guid, LocationInformationDto location)

  The function adds the location information to the emergency Connection with the specified GUID. All active call center connections are notified about the new information through callCenterController.PublishUpdatedActiveConnections.

- void MobileClientUpdatedPersonalInfo (string guid, PersonalInformationDto personal)

  The function updates the mobile emergency client user's personal information in the Connection with the specified GUID. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

- void MobileClientUpdatedMobileDeviceInfo (string guid, MobileDeviceInformationDto device)

  The function adds updated information about the mobile device to the emergency connection with the specified GUID. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

- void MobileClientUpdatedNoSoundStatus (string guid, bool noSound)

  The function sets the mobile emergency client request for an operation without sound to the Connection. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

- void MobileClientUpdatedConnectionPriority (string guid, ConnectionPriorityDto priority)

  The function adds updated connection priority to the emergency connection with the specified GUID. All active call center connections are notified about the connection priority change through callCenterController.PublishUpdated↩ ActiveConnections.

- void MobileClientUpdatedInstrumentList (string guid, List< MeasurementInstrumentDto > instruments)

  The dunction updates the list of measurument instruments configured to the emergency client with the specified connection GUID. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

- void MobileClientSentTextMessage (string guid, TextMessageDto textMessage)

  The function adds a new text based message to the connection message log. Only call center connections marked as handler for this connection are notified about the message through call to their pushUpdatedConnections method.

- void CallCenterClientSendTextMessage (CallCenterConnection callCenterConnection, ConnectionDto connection, TextMessageDto textMessage)

  The function sends a text message from a call center client to the specifeid mobile emergency client.

- void MobileClientUpdatedEmergencyType (string guid, EmergencyTypeDto emergencyType)

  The function updates the emergency type of an existing emergency connection instance to the specified value. Only the call center connections that are attached handlers for this connection are notified about the change.

- void MobileClientUpdatedMedicalInfo (string guid, MedicalInformationDto medicalInfo)

  The function handles medical information updates from mobile emergency client.

- List< ConnectionDto > getAllActiveConnections ()

  The function returns a List of all active emergency connections. Currently all connection instances in the Connection dictionary are returned.

- void CallCenterClientHasDisconnected (CallCenterConnection user)

The method handles call center client disconnect event. When a call center client disconnects, it no longer marked as a handler for any Connection. If it is the only active handler for a connection, the connection is transferred to hold state.

- void CallCenterClientOpenConnectionForProcessing (CallCenterConnection callCenterConnection, ConnectionDto connection)

  The function opens an emergency connection for handling. The provided CallCenterConnection is added to the active handlers of the provided connection unless it already is an associated handler for the connection. Request to open a processed connection for processing will throw a TargetStateFault.
- void CallCenterClientChangePriority (CallCenterConnection callCenterConnection, ConnectionDto connection)

  The function changes priority on the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.
- void CallCenterClientPutOpenOnHold (CallCenterConnection callCenterConnection, ConnectionDto connection)

  The function puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection. If there are other attached callers, the connection will not be put on hold and the requesting call center connection will only be removed from the handlers.
- void CallCenterClientMarkOpenProcessedClose (CallCenterConnection callCenterConnection, ConnectionDto connection)

  The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.
- void CallCenterClientRequestRemoteAction (CallCenterConnection callCenterConnection, ConnectionDto connection, RemoteActionDto action)

  The function requests an action to be performed by the provided remote mobile emergency client
- void CallCenterClientRequestMediaUpstreaming (CallCenterConnection callCenterConnection, ConnectionDto connection, MediaConfigurationDto mediaConfiguration)

  The function requests the mobile emergency client to start upstreaming media according to the provided configuration.
- void CallCenterClientRequestMediaDownstreaming (CallCenterConnection callCenterConnection, ConnectionDto connection, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.
- void CallCenterClientRequestStartMeasurement (CallCenterConnection callCenterConnection, ConnectionDto connection, MeasurementInstrumentDto measurementDevice)

  The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.
- void CallCenterClientRequestStopMeasurement (CallCenterConnection callCenterConnection, ConnectionDto connection, MeasurementInstrumentDto measurementDevice)

  The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.
- void MobileClientUpdatedConnectionLatencyInfo (string guid, ConnectionLatencyInformationDto latencyInfo)

  The function updates the latency information for the specified connection GUID.

Properties

- static ConnectionController Instance `[get]`

  The function returns the ConnectionController instance.

2.7.1 Detailed Description

The singleton class manages active emergency connections (Connection). Contains a dictionary for active connections and their identifying GUIDs. All emergency connection state and life cycle methods are handled by this class. It functions as a broker between the mobile emergency connection client and the call center client.

<author>Veli-Mikko Puupponen, Ilkka Rautiainen</author>

2.7.2 Member Function Documentation

2.7.2.1 void CallCenterClientChangePriority (CallCenterConnection *callCenterConnection,* ConnectionDto *connection*)

The function changes priority on the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

It Throws TargetStateFault if connection is processed, the requesting call center connection is not an associated handler or the connection to the mobile emergency client has already been disconnected. It Throws ConnectionFault if the provided Connection does not represent an open emergency connection.

Parameters

callCenterConnection	The CallCenterConnection changing the connection priority.
connection	The ConnectionDto identifying the mobile emergency client.

2.7.2.2 void CallCenterClientHasDisconnected (CallCenterConnection *user*)

The method handles call center client disconnect event. When a call center client disconnects, it no longer marked as a handler for any Connection. If it is the only active handler for a connection, the connection is transferred to hold state.

All call center clients are notified about handler configuration changes to the connections through the callCenterController.PublishUpdatedActiveConnections.

Parameters

user	The CallCenterConnection that has disconnected.

2.7.2.3 void CallCenterClientMarkOpenProcessedClose (CallCenterConnection *callCenterConnection,* ConnectionDto *connection*)

The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

It Throws TargetStateFault if connection is processed, the requesting call center connection is not an associated handler or the connection to the mobile emergency client has already been disconnected. It Throws ConnectionFault if the provided Connection does not represent an open emergency connection.

Parameters

callCenterConnection	The CallCenterConnection closing the connection.
connection	The ConnectionDto identifying the mobile emergency client.

be requested.

2.7.2.4 void CallCenterClientOpenConnectionForProcessing (CallCenterConnection *callCenterConnection,* ConnectionDto *connection*)

The function opens an emergency connection for handling. The provided CallCenterConnection is added to the active handlers of the provided connection unless it already is an associated handler for the connection. Request to open a processed connection for processing will throw a TargetStateFault.

If the Connection is not already in the InProgress state (i.e. no call center connection is currently associated as its handler), it is transferred to InPrograss state.

All call center clients are notified about handler configuration changes to the connections through the callCenterController.PublishUpdatedActiveConnections.

It Throws TargetStateFault if connection is processed or the requesting call center client is already an associated

handler for this connection. It Throws ConnectionFault if the provided Connection does not represent an open emergency connection.

Parameters

callCenter↩ Connection	The CallCenterConnection opening the connection for handling.
connection	The ConnectionDto for the Connection to be opened for handling.

2.7.2.5 void CallCenterClientPutOpenOnHold (CallCenterConnection *callCenterConnection,* ConnectionDto *connection*)

The function puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection. If there are other attached callers, the connection will not be put on hold and the requesting call center connection will only be removed from the handlers.

It Throws TargetStateFault if connection is processed, the requesting call center connection is not an associated handler or the connection to the mobile emergency client has already been disconnected. It Throws Connection↩ Fault if the provided Connection does not represent an open emergency connection.

Parameters

callCenter↩ Connection	The CallCenterConnection putting the connection on hold.
connection	The ConnectionDto identifying the mobile emergency client.

2.7.2.6 void CallCenterClientRequestMediaDownstreaming (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnection identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
mediaUrl	The location of the media to be played back at the mobile client.

2.7.2.7 void CallCenterClientRequestMediaUpstreaming (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* MediaConfigurationDto *mediaConfiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

<param name="callCenterConnection"The >CallCenterConnection identifying the existing call center client user connection.

Parameters

connection	The ConnectionDto identifying the target mobile emergency client connection.
media↩ Configuration	The quality parameters for the requested media.

2.7.2.8 void CallCenterClientRequestRemoteAction (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* RemoteActionDto *action*)

The function requests an action to be performed by the provided remote mobile emergency client

It Throws FaultException<ConnectionFault> if such connection does not currently exists in active connections collection. It Throws FaultException<TargetStateFault> if connection is already processed, has disconnected or the requester is not an associated handler for this connection.

Parameters

user	The CallCenterConnectionDto of the requester.
connection	The ConnectionDto of the target emergency connection.
action	The cation to be performed by the remote mobile emergency client.

2.7.2.9 void CallCenterClientRequestStartMeasurement (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnection identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device To be enabled and started uploading data from.

2.7.2.10 void CallCenterClientRequestStopMeasurement (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnection identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device to disable and stop uploading data from.

2.7.2.11 void CallCenterClientSendTextMessage (CallCenterConnection *callCenterConnection,* ConnectionDto *connection,* TextMessageDto *textMessage*)

The function sends a text message from a call center client to the specifeid mobile emergency client.

It Throws TargetStateFault if connection is processed, the requesting call center connection is not an associated handler or the connection to the mobile emergency client has already been disconnected. It Throws Connection↩ Fault if the provided Connection does not represent an open emergency connection.

Parameters

callCenter↩ Connection	The CallCenterConnection opening the connection for handling.

connection	The ConnectionDto for the Connection to be opened for handling.
textMessage	The text message to be sent to the mobile emergency client.

2.7.2.12 List<ConnectionDto> getAllActiveConnections ( )

The function returns a List of all active emergency connections. Currently all connection instances in the Connection dictionary are returned.

Returns

A list of all emergency connections.

2.7.2.13 string GetNewGuidStrig ( )

The function generates a new GUID. It currently uses .NET framework's Guid.NewGuid(). No collision checking is performed. All-zero GUIDs are not returned.

Returns

A new GUID formatted as a string.

2.7.2.14 string MobileClientConnected (string *transportId*)

The function handles connection from a new mobile emergency connection client. The client is given a GUID and marked to have arrived at System.DateTime.Now. The instantiated connection is set to be connected to a remote mobile emergency client. Priority of a new connection is always urgent.

The connection is added to the active connection dictionary and all active call center clients are informed about new connection through callCenterController.PublishUpdatedActiveConnections call.

Parameters

transportId	The SignalR connection id related to this client connection.

Returns

A guid assigned to this connection.

2.7.2.15 void MobileClientDisconnected (string *guid*)

The function handles disconnect event of a mobile emegency connection client. The Connection is marked as disconnected unless it already has the status "Processed". All active call center connections are notified about the connection state change through callCenterController.PublishUpdatedActiveConnections if the connection was not already in processed state.

Parameters

guid	The guid of an existing emergency connection.

2.7.2.16 void MobileClientReconnected (string *guid,* string *transportId*)

The function handles reconnection of a mobile emergency connection client. The SignalR connection id for this emergency connection is updated in the corresponding Connection instance.

Parameters

guid	The guid of an existing emergency connection.
transportId	The SignalR connection id related to this client connection.

2.7.2.17 void MobileClientSentTextMessage (string *guid,* TextMessageDto *textMessage*)

The function adds a new text based message to the connection message log. Only call center connections marked as handler for this connection are notified about the message through call to their pushUpdatedConnections method.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
textMessage	The text message from the mobile emergency client.

2.7.2.18 void MobileClientUpdatedConnectionLatencyInfo (string *guid,* ConnectionLatencyInformationDto *latencyInfo*)

The function updates the latency information for the specified connection GUID.

Parameters

guid	The guid of an existing emergency connection.
latencyInfo	the ltency information.

2.7.2.19 void MobileClientUpdatedConnectionPriority (string *guid,* ConnectionPriorityDto *priority*)

The function adds updated connection priority to the emergency connection with the specified GUID. All active call center connections are notified about the connection priority change through callCenterController.PublishUpdated↩ ActiveConnections.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
priority	The new priority information.

2.7.2.20 void MobileClientUpdatedEmergencyType (string *guid,* EmergencyTypeDto *emergencyType*)

The function updates the emergency type of an existing emergency connection instance to the specified value. Only the call center connections that are attached handlers for this connection are notified about the change.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
emergencyType	The emergency type specified by the mobile emergency client.

2.7.2.21 void MobileClientUpdatedInstrumentList (string *guid,* List< MeasurementInstrumentDto > *instruments*)

The dunction updates the list of measurument instruments configured to the emergency client with the specified connection GUID. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	GUID of an existing emergency connection.
instruments	The list of the instruments at the mobile device usin the connection with the GUID.

2.7.2.22 void MobileClientUpdatedLocation (string *guid,* LocationInformationDto *location*)

The function adds the location information to the emergency Connection with the specified GUID. All active call center connections are notified about the new information through callCenterController.PublishUpdatedActive↩ Connections.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
location	The emergency connection client's new location.

2.7.2.23 void MobileClientUpdatedMedicalInfo (string *guid,* MedicalInformationDto *medicalInfo*)

The function handles medical information updates from mobile emergency client.

TODO: this functionality is not implemented.

Parameters

guid	The guid of an existing emergency connection.
medicalInfo	The medican information for the emergency client.

2.7.2.24 void MobileClientUpdatedMobileDeviceInfo (string *guid,* MobileDeviceInformationDto *device*)

The function adds updated information about the mobile device to the emergency connection with the specified G↩ UID. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
device	The new device information.

2.7.2.25 void MobileClientUpdatedNoSoundStatus (string *guid,* bool *noSound*)

The function sets the mobile emergency client request for an operation without sound to the Connection. Only call center connections marked as handler for this connection are notified about the updated information through call to their pushUpdatedConnections method.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws Target↩ StateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
noSound	True, if operation without sound is requested, otherwise False.

2.7.2.26 void MobileClientUpdatedPersonalInfo (string *guid,* PersonalInformationDto *personal*)

The function updates the mobile emergency client user's personal information in the Connection with the specified GUID. Only call center connections marked as handler for this connection are notified about the updated information

through call to their pushUpdatedConnections method.

It Throws ConnectionFault if the provided GUID does not match an open emergency connection. It Throws TargetStateFault if the emergency connection with the provided GUID is already processed.

Parameters

guid	The guid of an existing emergency connection.
personal	The new personal information.

2.7.3 Property Documentation

2.7.3.1 ConnectionController Instance `[static],[get]`

The function returns the ConnectionController instance.

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

- ConnectionController.cs

2.8 ConnectionDatabaseContext Class Reference

The class defines database for the server.

Inherits DbContext.

Properties

- DbSet< Connection > **Connections** `[get, set]`
- DbSet< LocationInformation > **Locations** `[get, set]`
- DbSet< MobileDeviceInformation > **MobileDeviceInformations** `[get, set]`

2.8.1 Detailed Description

The class defines database for the server.

TODO: Not currently in use or up-to-date.

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

- ConnectionDatabaseContext.cs

2.9 ConnectionDto Class Reference

The dto container class corresponds to the Connection class.

Public Member Functions

- ConnectionDto ()

  The function initializes an empty ConnectionDto.

Properties

- string ConnectionGuid `[get, set]`

  The function gets or sets the Guid idenfying this connection in the system.

- DateTime ArrivalTime [get, set]

  The function gets or sets the DateTime specifying the moment in time when this connection first arrived to the server from the mobile emergency client.
- bool clientDisconnected [get, set]

  The function gets or sets the value indicating if the mobile emergency client repserensted by the instance has already disconnected its server connection. True, if the client has disconnected, otherwise false.
- bool RequestedNoSound [get, set]

  The function gets or sets the silent operation request state. True, if the mobile emergency client represented by the instance has requested operation without audio.
- ConnectionStateDto ConnectionState [get, set]

  The function gets or sets the state of this connection.
- PersonalInformationDto PersonalInformation [get, set]

  The function gets or sets the personal information provided by the mobile emergency client represented by this instace.
- ConnectionPriorityDto ConnectionPriority [get, set]

  The function gets or sets the priority of this emerngecy connection.
- EmergencyTypeDto EmergencyType [get, set]

  The function gets or sets the type of emergency provided by the mobile emergency client represented by the instance.
- List< LocationInformationDto > LocationInformation [get, set]

  The function gets or sets the collection containing all location information updates provided by the mobile emergency client during the emergency connection.
- List< MobileDeviceInformationDto > MobileDeviceInformation [get, set]

  The function gets or sets the collection containing all mobile device status information updates provided by the mobile emergency client during the emergency connection.
- List< CallCenterConnectionDto > AttachedCallCenterConnections [get, set]

  The function gets or sets the collection containing all CallCenterConnectionDto instances representing the call center clients that are currently handling this emergency connection.
- List< MeasurementInstrumentDto > MeasurementInstruments [get, set]

  Get sor sets the collection containing all MeasurementInstrumentDto instances provided on the mobile emergency client's last available measurement instrument list update.
- List< TextMessageDto > TextMessages [get, set]

  The function gets or sets the collection containing all text based messages received from or sent to the mobile emergency client during this emergency connection.

2.9.1 Detailed Description

The dto container class corresponds to the Connection class.

<author>Veli-Mikko Puupponen</author> It represents a mobile emergency client connection. It contains the GUID identifying the client, time stamp marking the connection arrival, current state of the connection and all information received from the client or sent to it during its connection.

2.9.2 Constructor & Destructor Documentation

2.9.2.1 ConnectionDto ( )

The function initializes an empty ConnectionDto.

2.9.3 Property Documentation

2.9.3.1 DateTime ArrivalTime [get],[set]

The function gets or sets the DateTime specifying the moment in time when this connection first arrived to the server from the mobile emergency client.

2.9.3.2 List<CallCenterConnectionDto> AttachedCallCenterConnections [get], [set]

The function gets or sets the collection containing all CallCenterConnectionDto instances representing the call center clients that are currently handling this emergency connection.

2.9.3.3 bool clientDisconnected [get], [set]

The function gets or sets the value indicating if the mobile emergency client repserensted by the instance has already disconnected its server connection. True, if the client has disconnected, otherwise false.

2.9.3.4 string ConnectionGuid [get], [set]

The function gets or sets the Guid idenfying this connection in the system.

2.9.3.5 ConnectionPriorityDto ConnectionPriority [get], [set]

The function gets or sets the priority of this emerngecy connection.

2.9.3.6 ConnectionStateDto ConnectionState [get], [set]

The function gets or sets the state of this connection.

2.9.3.7 EmergencyTypeDto EmergencyType [get], [set]

The function gets or sets the type of emergency provided by the mobile emergency client represented by the instance.

2.9.3.8 List<LocationInformationDto> LocationInformation [get], [set]

The function gets or sets the collection containing all location information updates provided by the mobile emergency client during the emergency connection.

2.9.3.9 List<MeasurementInstrumentDto> MeasurementInstruments [get], [set]

Get sor sets the collection containing all MeasurementInstrumentDto instances provided on the mobile emergency client's last available measurement instrument list update.

2.9.3.10 List<MobileDeviceInformationDto> MobileDeviceInformation [get], [set]

The function gets or sets the collection containing all mobile device status information updates provided by the mobile emergency client during the emergency connection.

2.9.3.11 PersonalInformationDto PersonalInformation [get], [set]

The function gets or sets the personal information provided by the mobile emergency client represented by this instace.

2.9.3.12 bool RequestedNoSound [get], [set]

The function gets or sets the silent operation request state. True, if the mobile emergency client represented by the instance has requested operation without audio.

2.9.3.13 List<TextMessageDto> TextMessages [get], [set]

The function gets or sets the collection containing all text based messages received from or sent to the mobile emergency client during this emergency connection.

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

- ConnectionDto.cs

2.10 ConnectionFault Class Reference

The fault class is used for exceptions related to connection state.

Inherits Fault.

Public Member Functions

- ConnectionFault (String cause, string detail)

  The function initializes a new ConnectionFault instance with the specified cause description and a detailed fault description.
- ConnectionFault (String cause)

  The function initializes a new ConnectionFault instance with the specified cause description.

Additional Inherited Members

2.10.1 Detailed Description

The fault class is used for exceptions related to connection state.

<author>Veli-Mikko Puupponen</author>

2.10.2 Constructor & Destructor Documentation

2.10.2.1 ConnectionFault (String *cause*, string *detail*)

The function initializes a new ConnectionFault instance with the specified cause description and a detailed fault description.

Parameters

cause	The description of the cause.
detail	The more detailed description of the exception.

2.10.2.2 ConnectionFault (String *cause*)

The function initializes a new ConnectionFault instance with the specified cause description.

Parameters

cause	The description of the cause.

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

- ConnectionFault.cs

2.11 ConnectionLatencyInformation Class Reference

Properties

- int Id `[get, set]`

  Class for emergency mobile client's connection latency information.
- int **ConnectionId** `[get, set]`
- long[] **roundtripTimeMs** `[get, set]`
- int **uplinkTestPacketBytes** `[get, set]`
- int **downlinkTestPacketBytes** `[get, set]`
- double **uplinkSpeedEstimateMbPerSecond** `[get, set]`

- double **downlinkSpeedEstimateMbPerSecond** `[get, set]`
- int **numberOfTests** `[get, set]`
- int **smallTestPacketSizeBytes** `[get, set]`
- int **bigTestPacketSizeBytes** `[get, set]`

2.11.1 Property Documentation

2.11.1.1 int Id `[get],[set]`

Class for emergency mobile client's connection latency information.

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

- ConnectionLatencyInformation.cs

2.12 ConnectionLatencyInformationDto Class Reference

The dto container class corresponds to the ConnectionLatencyInformation class.

Properties

- long[] roundtripTimeMs `[get, set]`

  The function gets or sets the round trip time in milliseconds. It is used to track the time used on transfer.
- int uplinkTestPacketBytes `[get, set]`

  The function gets or sets the size of the uplink test packet in bytes.
- int downlinkTestPacketBytes `[get, set]`

  The function gets or sets the size of the downlink test packet in bytes.
- double uplinkSpeedEstimateMbPerSecond `[get, set]`

  The function gets or sets the estimated uplink speed in megabytes per second.
- double downlinkSpeedEstimateMbPerSecond `[get, set]`

  The function gets or sets the estimated downlink speed in megabytes per second.
- int numberOfTests `[get, set]`

  The function gets or sets the number of the test to complete.
- int smallTestPacketSizeBytes `[get, set]`

  The function gets or sets the size of the small test packet in bytes.
- int bigTestPacketSizeBytes `[get, set]`

  The function gets or sets the size of the big test packet in bytes.

2.12.1 Detailed Description

The dto container class corresponds to the ConnectionLatencyInformation class.

<author>Ilkka Rautiainen</author> The class represents the latency in the connections. It contains the uplink and downlink test packets, the estimated upload and download speeds, number of the tests and the time to travel.

2.12.2 Property Documentation

2.12.2.1 int bigTestPacketSizeBytes `[get],[set]`

The function gets or sets the size of the big test packet in bytes.

2.12.2.2 double downlinkSpeedEstimateMbPerSecond `[get],[set]`

The function gets or sets the estimated downlink speed in megabytes per second.

2.12.2.3 int downlinkTestPacketBytes [get],[set]

The function gets or sets the size of the downlink test packet in bytes.

2.12.2.4 int numberOfTests [get],[set]

The function gets or sets the number of the test to complete.

2.12.2.5 long [] roundtripTimeMs [get],[set]

The function gets or sets the round trip time in milliseconds. It is used to track the time used on transfer.

2.12.2.6 int smallTestPacketSizeBytes [get],[set]

The function gets or sets the size of the small test packet in bytes.

2.12.2.7 double uplinkSpeedEstimateMbPerSecond [get],[set]

The function gets or sets the estimated uplink speed in megabytes per second.

2.12.2.8 int uplinkTestPacketBytes [get],[set]

The function gets or sets the size of the uplink test packet in bytes.

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

- ConnectionLatencyInformationDto.cs

2.13 DatabaseController Class Reference

The class is for database interaction. It handles database connections and operations for storing and retrieving Connection-instances for persistency. It uses EntityFramework.

Public Member Functions

- void ReinitializeDatabase ()

  The function reinitializes the database connection.
- Connection InitializeNewConnection ()

  The function initializes a new connection and adds it to the database. It is used when a new client connects to the server.
- void UpdateConnection (Connection co)

  The function to update the connection in the database. TODO: Update the function to accept also a new connection
- Connection RetrieveConnection (int key)

  The function to retrieve a connection specified with the key from the database.

Properties

- static DatabaseController Instance [get]

  The function to get the instance of DatabaseController.

2.13.1 Detailed Description

The class is for database interaction. It handles database connections and operations for storing and retrieving Connection-instances for persistency. It uses EntityFramework.

<author>Veli-Mikko Puupponen</author> TODO: Not currently in use or up-to-date.

2.13.2 Member Function Documentation

2.13.2.1 Connection InitializeNewConnection ( )

The function initializes a new connection and adds it to the database. It is used when a new client connects to the server.

Returns

A new connection.

2.13.2.2 void ReinitializeDatabase ( )

The function reinitializes the database connection.

2.13.2.3 Connection RetrieveConnection (int *key*)

The function to retrieve a connection specified with the key from the database.

Parameters

key	The guid of the connection.

Returns

The requested connection.

2.13.2.4 void UpdateConnection (Connection *co*)

The function to update the connection in the database. TODO: Update the function to accept also a new connection

Parameters

co	The connection to be updated.

2.13.3 Property Documentation

2.13.3.1 DatabaseController Instance [static],[get]

The function to get the instance of DatabaseController.

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

- DatabaseController.cs

2.14 DataTransferController Class Reference

The class is for managing audio/video and measurement data distribution from the mobile emergency clients to the associated call center clients. The data packets from the mobile emergency clients can be relayed in a fan-out style to multiple call center connections. The data subscriptions for each call center connection are stored in the "dataSubscribers" dictionary.

Public Member Functions

- void Disable ()

  The function disables the UdpMediaRelayServerCore to release its undelying UDP socket.
- void SubscribeCallCenterClientForData (CallCenterConnection callCenter, Connection mobileClient)

The function adds the call center client as a receiver for audio/video and instrument measurement data sent by the specified mobile emergency client. Also adds the call center client as the sole sender of data to the mobile emerngenct client.

- void UnsubscribeCallCenterClientForData (CallCenterConnection callCenter, Connection mobileClient)

  The function unsubscribes the call center client from audio/video and instrument measurement data sent by the specified mobile emergency client.

- void UnsubscribeCallCenterClientForAnyData (CallCenterConnection callCenter)

  The function unsubscribes the call center client from audio/video and instrument measurement data sent by any mobile emergency client. Also removes queues and mappings for publishing data from the call center client to any mobile emergency client.

- void PublishMobileClientAudioVideo (string mobileGuid, MediaInformationDto mediaInformation, byte[] mediaPayload)

  The function performs fan-out distribution of audio/video data received from a mobile emergency client with the specified GUID. The data is distributed to the call center clients that have subscribed for data from mobile client using the specified GUID.

- bool PublishCallCenterClientAudioVideo (CallCenterConnectionDto user, MediaInformationDto mediaInfo, byte[] mediaData)

  The function publishes audio and video from a call center client to its client specific queue. This data can be asynchronously read by the remote mobile emergency client over its WCF connection.

- AudioVideoContainerDto MobileClientGetAudioVideoFromPublishQueue (string mobileGuid)

  The function returns audio and video media published by a call center connection. If the publishing queue for the call center client associated with the calling mobile emergency client, this method returns an empty AudioVideo↩ ContainerDto. Otherwise returns the AudioVideoContainerDto from the media queue.

- void PublishMobileClientInstrumentData (string mobileGuid, MeasurementInstrumentDto measurement↩ Instrument, byte[] measurementPayload)

  The function performs fan-out distribution of the measurement data received from a mobile emergency client with the specified GUID. The data is distributed to the call center clients that have subscribed for data from mobile client using the specified GUID.

Properties

- static DataTransferController Instance `[get]`

  The function returns the instanfe of DatatransferController.

2.14.1 Detailed Description

The class is for managing audio/video and measurement data distribution from the mobile emergency clients to the associated call center clients. The data packets from the mobile emergency clients can be relayed in a fan-out style to multiple call center connections. The data subscriptions for each call center connection are stored in the "dataSubscribers" dictionary.

<author>Veli-Mikko Puupponen</author> The data packets from call center clients to the mobile emergency clients are enqueued in a call center client specific queue from which the mobile emergency client that the call center client is currently processing is able to fetch them.

Contains an instance of UdpMediaRelayServerCore that manages the UDP based media data transfer.

TODO: Contains a desctuctor to release the UDP socket used by the UdpMediaRelayServerCore. This should be handled by the life-time events of the service. Global.asax.cs contains methods Application_Start and Application↩ _End, but they are not invoked if the service instance is loaded by the WAS as a result of a call to the TCPBinding used by the CallCenterService. One possible solution would be implementing a HostFactory. For more information, see article at http://msdn.microsoft.com/en-us/magazine/cc163357.aspx and the code in "↩ Figure 11".

2.14.2 Member Function Documentation

2.14.2.1 void Disable ( )

The function disables the UdpMediaRelayServerCore to release its undelying UDP socket.

2.14.2.2 AudioVideoContainerDto MobileClientGetAudioVideoFromPublishQueue (string *mobileGuid*)

The function returns audio and video media published by a call center connection. If the publishing queue for the call center client associated with the calling mobile emergency client, this method returns an empty AudioVideo↩ ContainerDto. Otherwise returns the AudioVideoContainerDto from the media queue.

Parameters

mobileGuid	The guid of an existing emergency mobile client that is connected to the server.

Returns

AudioVideoContainerDto that constains audio/video data, if available. Otherwise it is empty.

2.14.2.3 bool PublishCallCenterClientAudioVideo (CallCenterConnectionDto *user,* MediaInformationDto *mediaInfo,* byte[ ] *mediaData*)

The function publishes audio and video from a call center client to its client specific queue. This data can be asynchronously read by the remote mobile emergency client over its WCF connection.

Parameters

user	The CallCenterConnectionDto describing the publishing call center client.
mediaInfo	The MediaInformationDto describing the published media data.
mediaData	The media data.

Returns

Bolean whether the publishing succseeded or not.

2.14.2.4 void PublishMobileClientAudioVideo (string *mobileGuid,* MediaInformationDto *mediaInformation,* byte[ ] *mediaPayload*)

The function performs fan-out distribution of audio/video data received from a mobile emergency client with the specified GUID. The data is distributed to the call center clients that have subscribed for data from mobile client using the specified GUID.

Parameters

mobileGuid	The guid of the mobile emergency client connection providing the audio/video.
media↩ Information	The description of the audio/video data.
mediaPayload	The audio/video payload data.

2.14.2.5 void PublishMobileClientInstrumentData (string *mobileGuid,* MeasurementInstrumentDto *measurementInstrument,* byte[ ] *measurementPayload*)

The function performs fan-out distribution of the measurement data received from a mobile emergency client with the specified GUID. The data is distributed to the call center clients that have subscribed for data from mobile client using the specified GUID.

Parameters

mobileGuid	The guid of the mobile emergency client connection providing the data.
measurement↩ Instrument	The instrument producing the measurement data.
measurement↩ Payload	The measurement data from the instrument.

2.14.2.6 void SubscribeCallCenterClientForData (CallCenterConnection *callCenter,* Connection *mobileClient*)

The function adds the call center client as a receiver for audio/video and instrument measurement data sent by the specified mobile emergency client. Also adds the call center client as the sole sender of data to the mobile emerngenct client.

Parameters

callCenter	The subscribing call center client.
mobileClient	The mobile emergency client from whon the data is subscribed from.

2.14.2.7 void UnsubscribeCallCenterClientForAnyData (CallCenterConnection *callCenter*)

The function unsubscribes the call center client from audio/video and instrument measurement data sent by any mobile emergency client. Also removes queues and mappings for publishing data from the call center client to any mobile emergency client.

Parameters

callCenter	The unsubscribing call center client.

2.14.2.8 void UnsubscribeCallCenterClientForData (CallCenterConnection *callCenter,* Connection *mobileClient*)

The function unsubscribes the call center client from audio/video and instrument measurement data sent by the specified mobile emergency client.

Parameters

callCenter	The unsubscribing call center client.
mobileClient	The mobile emergency client from whom data is no more subscribed by the specified call center connection.

2.14.3 Property Documentation

2.14.3.1 DataTransferController Instance `[static],[get]`

The function returns the instanfe of DatatransferController.

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

- DataTransferController.cs

2.15 EmergencyType Class Reference

Class for storing the descriptive emergency type information provided by the mobile emergency client.

Properties

- string TypeName `[get, set]`

  The function gets or sets the string description of the emergency type.

2.15.1 Detailed Description

Class for storing the descriptive emergency type information provided by the mobile emergency client.

<author>Veli-Mikko Puupponen</author> It currently stores only a string representation of the information.

2.15.2 Property Documentation

2.15.2.1 string TypeName [get],[set]

The function gets or sets the string description of the emergency type.

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

- EmergencyType.cs

2.16 EmergencyTypeDto Class Reference

The dto class corresponds to the EmergencyType class.

Properties

- string TypeName [get, set]
  The function to get and set the description of the emergency type.

2.16.1 Detailed Description

The dto class corresponds to the EmergencyType class.

<author>Veli-Mikko Puupponen</author> It is a container class for storing descriptive emergency type information provided by the mobile emergency client.

It currently stores only a string representation of the information.

2.16.2 Property Documentation

2.16.2.1 string TypeName [get],[set]

The function to get and set the description of the emergency type.

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

- EmergencyTypeDto.cs

2.17 Fault Class Reference

The base class is used for all fault exceptions transferred through the WCF interfaces of the server.

Inherited by ConnectionFault, ParameterFault, and TargetStateFault.

Public Member Functions

- Fault (String cause, String detail)
  The function initializes a new Fault instance with the specified cause description and a detailed fault description.
- Fault (String cause)
  The function initializes a new Fault instance with the specified cause description.

Properties

- string Cause [get, set]

  Short description of the cause of the exception.
- string Detail [get, set]

  Detailed description of the exception.

2.17.1 Detailed Description

The base class is used for all fault exceptions transferred through the WCF interfaces of the server.

<author>Veli-Mikko Puupponen</author>

2.17.2 Constructor & Destructor Documentation

2.17.2.1 Fault (String *cause*, String *detail*)

The function initializes a new Fault instance with the specified cause description and a detailed fault description.

Parameters

cause	The description of the cause.
detail	The more detailed description of the exception.

2.17.2.2 Fault (String *cause*)

The function initializes a new Fault instance with the specified cause description.

Parameters

cause	The description of the cause.

2.17.3 Property Documentation

2.17.3.1 string Cause [get],[set]

Short description of the cause of the exception.

2.17.3.2 string Detail [get],[set]

Detailed description of the exception.

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

- Fault.cs

2.18 Global Class Reference

The class is generated code. Server uses it. Do not modify.

Inherits HttpApplication.

Protected Member Functions

- void **Application_Start** (object sender, EventArgs e)
- void **Session_Start** (object sender, EventArgs e)
- void **Application_BeginRequest** (object sender, EventArgs e)

- void **Application_AuthenticateRequest** (object sender, EventArgs e)
- void **Application_Error** (object sender, EventArgs e)
- void **Session_End** (object sender, EventArgs e)
- void **Application_End** (object sender, EventArgs e)

2.18.1 Detailed Description

The class is generated code. Server uses it. Do not modify.

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

- Global.asax.cs

2.19 IMobileClientMethods Interface Reference

The interface defines all callback methods supported by the mobile emergency client's SignalR client.

Inherited by MobileClientController.

Public Member Functions

- void RequestLocationUpdate (String transportId)

  The function sends location information update request to the emergency client with the specified SignalR connection ID.
- void RequestDeviceInfo (String transportId)

  The function sends mobile device information update request to the emergency client with the specified SignalR connection ID.
- void UpdateConnectionStatus (String transportId, ConnectionStateDto status)

  The function sends an update connection status to the emergency client with the specified SignalR connection ID.
- void RequestUserInfo (String transportId)

  The function sends user information update request to the emergency client with the specified SignalR connection ID.
- void RequestMedicalInfo (String transportId)

  The function sends medical information update request to the emergency client with the specified SignalR connection ID.
- void DisplayUserLocationMap (String transportId)

  Orders the mobile emergency client to display a map enabling the user to manually specify the current location.
- void CloseUserLocationMap (String transportId)

  Orders the mobile emergency client to hide the location map.
- void GetInstrumentList (String transportId)

  The function requests a list of available measurement instruments from the emergency client with the specified SignalR connection ID.
- void RequestMediaUpstreaming (String transportId, MediaConfigurationDto mediaCofiguration)

  The function requests the mobile emergency client to start upstreaming media according to the provided configuration
- void RequestMediaDownstreaming (String transportId, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and displaying media from the provided location.
- void RequestStartMeasurement (String transportId, MeasurementInstrumentDto instrument)

  The function requests the mobile emergency client with the specifiedSignalR connection ID to start measuring using the specified instrument and uploading the measurement data.
- void RequestStopMeasurement (String transportId, MeasurementInstrumentDto instrument)

  The function requests the mobile emergency client with the specified SignalR connection ID to stop measurement with the specified instrument and to no longer upload measurement data from it.
- void IncomingTextMessage (String transportId, TextMessageDto textMessage)

The function sends a text based message the mobile emergency client with the specified SignalR connection ID.

- void RequestConnectionLatencyInfo (String transportId)

  The function requests the mobile emergency client with the specified SignalR connection ID to start connection latency measurement.

2.19.1 Detailed Description

The interface defines all callback methods supported by the mobile emergency client's SignalR client.

<author>Veli-Mikko Puupponen</author>

2.19.2 Member Function Documentation

2.19.2.1 void CloseUserLocationMap (String *transportId*)

Orders the mobile emergency client to hide the location map.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.2 void DisplayUserLocationMap (String *transportId*)

Orders the mobile emergency client to display a map enabling the user to manually specify the current location.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.3 void GetInstrumentList (String *transportId*)

The function requests a list of available measurement instruments from the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.4 void IncomingTextMessage (String *transportId,* TextMessageDto *textMessage*)

The function sends a text based message the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
textMessage	The text message to be sent to the mobile emergency client.

2.19.2.5 void RequestConnectionLatencyInfo (String *transportId*)

The function requests the mobile emergency client with the specified SignalR connection ID to start connection latency measurement.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.6 void RequestDeviceInfo (String *transportId*)

The function sends mobile device information update request to the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.7 void RequestLocationUpdate (String *transportId*)

The function sends location information update request to the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.8 void RequestMediaDownstreaming (String *transportId,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and displaying media from the provided location.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
mediaUrl	The location of the media to be displayed by the mobile emergency client.

Implemented in MobileClientController.

2.19.2.9 void RequestMediaUpstreaming (String *transportId,* MediaConfigurationDto *mediaCofiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
media↩ Cofiguration	The configuration describing the requested media.

Implemented in MobileClientController.

2.19.2.10 void RequestMedicalInfo (String *transportId*)

The function sends medical information update request to the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.11 void RequestStartMeasurement (String *transportId,* MeasurementInstrumentDto *instrument*)

The function requests the mobile emergency client with the specifiedSignalR connection ID to start measuring using the specified instrument and uploading the measurement data.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
instrument	The target measurement instrument.

Implemented in MobileClientController.

2.19.2.12 void RequestStopMeasurement (String *transportId,* MeasurementInstrumentDto *instrument*)

The function requests the mobile emergency client with the specified SignalR connection ID to stop measurement with the specified instrument and to no longer upload measurement data from it.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
instrument	The target measurement instrument.

Implemented in MobileClientController.

2.19.2.13 void RequestUserInfo (String *transportId*)

The function sends user information update request to the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implemented in MobileClientController.

2.19.2.14 void UpdateConnectionStatus (String *transportId,* ConnectionStateDto *status*)

The function sends an update connection status to the emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
status	The new connection state.

Implemented in MobileClientController.

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

- IMobileClientMethods.cs

2.20 IWcfCallCenterCallback Interface Reference

The interface defines the callback methods for the call center clients.

Public Member Functions

- IAsyncResult BeginActiveConnectionsUpdated (List< ConnectionDto > updatedConnections, AsyncCallback callback, object asyncState)

  The asynchronous function callback method starts publishing the list of updated Connections to the associated call center connection.
- void EndActiveConnectionsUpdated (IAsyncResult result)

  The function to end the publishing of the list of updated Connections. The function is triggered after the asynchronous function has ended.
- IAsyncResult BeginAudioVideoReceived (string sourceGuid, MediaInformationDto mediaInfo, byte[] mediaData, AsyncCallback callback, object asyncState)

  The asynchronous function callback method to send audio/video from mobile emergency client to the associated call center connection.

- void EndAudioVideoReceived (IAsyncResult result)

  The function to end the sending if the audio/video. The function is triggered after the asynchronous function has ended.
- IAsyncResult BeginMeasurementDataReceived (string sourceGuid, MeasurementInstrumentDto instrument, byte[] measurementData, AsyncCallback callback, object asyncState)

  The asynchronous function callback method to send measurement data from mobile emergency client to the associated call center connection.
- void EndMeasurementDataReceived (IAsyncResult result)

  The function to end the sending of the measurement data. The function is triggered after the asynchronous function has ended.

2.20.1 Detailed Description

The interface defines the callback methods for the call center clients.

2.20.2 Member Function Documentation

2.20.2.1 IAsyncResult BeginActiveConnectionsUpdated (List< ConnectionDto > *updatedConnections,* AsyncCallback *callback,* object *asyncState*)

The asynchronous function callback method starts publishing the list of updated Connections to the associated call center connection.

Parameters

updatedConnections	The list of ConnectionDtos.
callback	The callback to be called when operation finishes.
asyncState	The state of the asynchronous function callbak.

Returns

The result from the asyncrhonous callback.

2.20.2.2 IAsyncResult BeginAudioVideoReceived (string *sourceGuid,* MediaInformationDto *mediaInfo,* byte[] *mediaData,* AsyncCallback *callback,* object *asyncState*)

The asynchronous function callback method to send audio/video from mobile emergency client to the associated call center connection.

Parameters

sourceGuid	The guid of the mobile emergency client sending the data.
mediaInfo	The description of the media.
mediaData	Media data.
callback	The callback to be called when operation finishes.
asyncState	The state of the asynchronous function callbak.

Returns

The result from the asyncrhonous callback.

2.20.2.3 IAsyncResult BeginMeasurementDataReceived (string *sourceGuid,* MeasurementInstrumentDto *instrument,* byte[] *measurementData,* AsyncCallback *callback,* object *asyncState*)

The asynchronous function callback method to send measurement data from mobile emergency client to the associated call center connection.

Parameters

sourceGuid	The guid of the mobile emergency client sending the data.
instrument	The instrument providing the measurement data.
measurement↩ Data	The masurement data fragment.
callback	The callback to be called when operation finishes.
asyncState	The state of the asynchronous function callbak.

Returns

The result from the asyncrhonous callback.

2.20.2.4 void EndActiveConnectionsUpdated (IAsyncResult *result*)

The function to end the publishing of the list of updated Connections. The function is triggered after the asynchronous function has ended.

Parameters

result	The result from the asynchronous function callback to publish the list of the updated Connections.

2.20.2.5 void EndAudioVideoReceived (IAsyncResult *result*)

The function to end the sending if the audio/video. The function is triggered after the asynchronous function has ended.

Parameters

result	The result from the asynchronous function callback to send audio/video.

2.20.2.6 void EndMeasurementDataReceived (IAsyncResult *result*)

The function to end the sending of the measurement data. The function is triggered after the asynchronous function has ended.

Parameters

result	The result from the asynchronous function callback to send the measurement data.

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

- IWcfCallCenterService.cs

2.21 IWcfCallCenterService Interface Reference

The interface defines the WCF service for the call center clients.

Inherited by WcfCallCenterService.

Public Member Functions

- CallCenterConnectionDto Connect (UserCredentialsDto credentials)

  The method handles call center client connections. This method has to be invoked by all call center clients before any operations on the server can be performed.
- void Reconnect (CallCenterConnectionDto user)

  The method handles call center client reconnections.
- void Disconnect (CallCenterConnectionDto user)

The method handles call center client disconnects. Every call center client should invoke this method before closing connection.

- List< ConnectionDto > GetActiveConnections (CallCenterConnectionDto user)

  The function requests a complete list of all mobile emergency client connections currently active on the server.

- void OpenConnectionForProcessing (CallCenterConnectionDto user, ConnectionDto connection)

  The function opens a mobile emergency client connection for handling in this call center connection.

- void TransferConnection (CallCenterConnectionDto user, List< CallCenterConnectionDto > targetCall↩ CenterConnections)

  The function transfers a mobile emergency client connection from the requesting call center connection to the specified call center connection(s).

- void SetConnectionPriority (CallCenterConnectionDto user, ConnectionDto connection)

  The function changes the priority for the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

- void MoveConnectionToHold (CallCenterConnectionDto user, ConnectionDto connection)

  Puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.

- void MarkProcessedCloseConnection (CallCenterConnectionDto user, ConnectionDto connection)

  The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

- void RequestRemoteAction (CallCenterConnectionDto user, ConnectionDto connection, RemoteActionDto action)

  The function requests an operation with no parameters to be executed by the provided mobile emergency connection.

- void RequestMediaUpstreaming (CallCenterConnectionDto user, ConnectionDto connection, Media↩ ConfigurationDto mediaConfiguration)

  The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

- void RequestMediaDownstreaming (CallCenterConnectionDto user, ConnectionDto connection, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

- void RequestStartMeasurement (CallCenterConnectionDto user, ConnectionDto connection, Measurement↩ InstrumentDto measurementDevice)

  The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

- void RequestStopMeasurement (CallCenterConnectionDto user, ConnectionDto connection, Measurement↩ InstrumentDto measurementDevice)

  The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

- void SendTextMessage (CallCenterConnectionDto user, ConnectionDto connection, TextMessageDto text↩ Message)

  v The function sends a text based message to the specified mobile emergency client.

- bool UploadMediaSegment (CallCenterConnectionDto user, MediaInformationDto mediaInfo, byte[] media↩ Data)

  The function uploads a segment of media from the call center client to the server. The media is forwarded to the mobile emergency client attached to the connection that the call center client currently is handling, if any.

- int Ping (int pingSequence)

  The function to ping the client. It is used to determine whether the client is still connected or not.

2.21.1 Detailed Description

The interface defines the WCF service for the call center clients.

<author>Veli-Mikko Puupponen</author>

2.21.2 Member Function Documentation

2.21.2.1 CallCenterConnectionDto Connect (UserCredentialsDto *credentials*)

The method handles call center client connections. This method has to be invoked by all call center clients before any operations on the server can be performed.

Parameters

credentials	The call center client login credentials.

Returns

The CallCenterConnectionDto used to identify the call center client in subsequent operations.

Implemented in WcfCallCenterService.

2.21.2.2 void Disconnect (CallCenterConnectionDto *user*)

The method handles call center client disconnects. Every call center client should invoke this method before closing connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Implemented in WcfCallCenterService.

2.21.2.3 List<ConnectionDto> GetActiveConnections (CallCenterConnectionDto *user*)

The function requests a complete list of all mobile emergency client connections currently active on the server.

It Throws ConnectionFault if the provided CallCenterConnection is not a valid active connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Returns

A list of active emergency connections.

Implemented in WcfCallCenterService.

2.21.2.4 void MarkProcessedCloseConnection (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.

Implemented in WcfCallCenterService.

2.21.2.5 void MoveConnectionToHold (CallCenterConnectionDto *user,* ConnectionDto *connection*)

Puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the mobile emergency client.

Implemented in WcfCallCenterService.

2.21.2.6 void OpenConnectionForProcessing (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function opens a mobile emergency client connection for handling in this call center connection.

It Throws ConnectionFault, if the provided CallCenterConnectionDto is invalid It Throws TargetStateFault if connection is already processed, does not exist or the requesting call center client is already an associated handler for this connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto for the connection to be opened for handling.

Implemented in WcfCallCenterService.

2.21.2.7 int Ping (int *pingSequence*)

The function to ping the client. It is used to determine whether the client is still connected or not.

Parameters

pingSequence	The sequence number of the ping.

Returns

A new ping with a new sequence number.

Implemented in WcfCallCenterService.

2.21.2.8 void Reconnect (CallCenterConnectionDto *user*)

The method handles call center client reconnections.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Implemented in WcfCallCenterService.

2.21.2.9 void RequestMediaDownstreaming (CallCenterConnectionDto *user,* ConnectionDto *connection,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
mediaUrl	The location of the media to be played back at the mobile client.

Implemented in WcfCallCenterService.

2.21.2.10 void RequestMediaUpstreaming (CallCenterConnectionDto *user*, ConnectionDto *connection*, MediaConfigurationDto *mediaConfiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
mediaConfiguration	The quality parameters for the requested media.

Implemented in WcfCallCenterService.

2.21.2.11 void RequestRemoteAction (CallCenterConnectionDto *user*, ConnectionDto *connection*, RemoteActionDto *action*)

The function requests an operation with no parameters to be executed by the provided mobile emergency connection.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
action	The request to be executed on the remote call center client.

Implemented in WcfCallCenterService.

2.21.2.12 void RequestStartMeasurement (CallCenterConnectionDto *user*, ConnectionDto *connection*, MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurementDevice	The measurement device To be enabled and started uploading data from.

Implemented in WcfCallCenterService.

2.21.2.13 void RequestStopMeasurement (CallCenterConnectionDto *user*, ConnectionDto *connection*, MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device to disable and stop uploading data from.

Implemented in WcfCallCenterService.

2.21.2.14 void SendTextMessage (CallCenterConnectionDto *user,* ConnectionDto *connection,* TextMessageDto *textMessage*)

v The function sends a text based message to the specified mobile emergency client.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
textMessage	The test message to be sent to the specified emergency client.

Implemented in WcfCallCenterService.

2.21.2.15 void SetConnectionPriority (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function changes the priority for the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection thas is a handler for the connection to be modified.

.

Parameters

connection	The ConnectionDto identifying the mobile emergency client.

Implemented in WcfCallCenterService.

2.21.2.16 void TransferConnection (CallCenterConnectionDto *user,* List< CallCenterConnectionDto > *targetCallCenterConnections*)

The function transfers a mobile emergency client connection from the requesting call center connection to the specified call center connection(s).

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection thas is a handler for the connection to be transferred.

.

Parameters

targetCall↩ Center↩ Connections	The CallCenterConnectionDtos identifying the transfer targets.

Implemented in WcfCallCenterService.

2.21.2.17 bool UploadMediaSegment (CallCenterConnectionDto *user,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function uploads a segment of media from the call center client to the server. The media is forwarded to the mobile emergency client attached to the connection that the call center client currently is handling, if any.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws TargetStateFault if the connection is processed or not a connection that the invoking call center connection is processing.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
mediaInfo	The description of the media.
mediaData	The media data bytes.

Implemented in WcfCallCenterService.

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

- IWcfCallCenterService.cs

2.22 IWcfMobileService Interface Reference

The interface defines the WCF service for the emergency mobile clients. The interface is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.

Inherited by WcfMobileService.

Public Member Functions

- void UpdateLocation (string guid, LocationInformationDto location)

  The function sends a new location information to the server.
- void UpdateDeviceInfo (string guid, MobileDeviceInformationDto deviceInfo)

  The function sends a new mobile device status information to the server.
- void UpdatePersonalInfo (string guid, PersonalInformationDto userInfo)

  The function sends a new mobile emergency client user's information to the server.
- void UpdateMedicalInfo (string guid, MedicalInformationDto medicalInfo)

  The function sends a new mobile emergency client user medical information to the server.
- void UpdateConnectionPriority (string guid, ConnectionPriorityDto priority)

  The function sends the selected emergency priority to the server.
- void UpdateRequestType (string guid, EmergencyTypeDto requestType)

  The function sends the selected emergency type to the server.
- void ToggleNoSound (string guid, bool noSound)

  The function sets mobile emergency client request for an operation without sound.
- void UpdateInstrumentList (string guid, List< MeasurementInstrumentDto > instruments)

  Updates the list of supported measurement instruments to the server.
- void UploadMediaSegment (string guid, MediaInformationDto mediaInfo, byte[] mediaData)

  The function uploads a segment of media from the mobile emergency client to the server.
- AudioVideoContainerDto GetMediaSegment (string guid)

  The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.
- int Ping (int pingSequence)

  The function to ping the client. It is used to determine whether the client is still connected or not.

- void UploadMeasurementData (string guid, MeasurementInstrumentDto instrument, byte[] measurement↩ Data)

  The function uploads a segment of the measurement data from the instrument at the mobile device to the server.
- void SendTextMessage (string guid, TextMessageDto textMessage)

  The function sends a text based message to the call center client handling the emergency connection.
- byte[] SendTestPacket (string guid, byte[] testPacket)

  The function sends a test packet to the server.
- void UpdateConnectionLatencyInfo (string guid, ConnectionLatencyInformationDto latencyInfo)

  The function updates the connection latency information to the server.

2.22.1 Detailed Description

The interface defines the WCF service for the emergency mobile clients. The interface is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.

<author>Veli-Mikko Puupponen</author>

2.22.2 Member Function Documentation

2.22.2.1 AudioVideoContainerDto GetMediaSegment (string *guid*)

The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.

Parameters

guid	The guid identifying the emergency client.

Returns

A AudioVideoContainerDto containing audio/video from the call center client. If no media is available, Audio↩ VideoContainerDto is empty with no payload.

Implemented in WcfMobileService.

2.22.2.2 int Ping (int *pingSequence*)

The function to ping the client. It is used to determine whether the client is still connected or not.

Parameters

pingSequence	The sequence number of the ping.</p

Implemented in WcfMobileService.

2.22.2.3 byte [] SendTestPacket (string *guid,* byte[] *testPacket*)

The function sends a test packet to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
testPacket	The test packet.

Returns

A byte array of test data.

Implemented in WcfMobileService.

2.22.2.4 void SendTextMessage (string *guid,* TextMessageDto *textMessage*)

The function sends a text based message to the call center client handling the emergency connection.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
textMessage	The text message to the call center.

Implemented in WcfMobileService.

2.22.2.5 void ToggleNoSound (string *guid,* bool *noSound*)

The function sets mobile emergency client request for an operation without sound.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
noSound	True if the operation without sound is requested, otherwise False.

Implemented in WcfMobileService.

2.22.2.6 void UpdateConnectionLatencyInfo (string *guid,* ConnectionLatencyInformationDto *latencyInfo*)

The function updates the connection latency information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
latencyInfo	Latency information.

Implemented in WcfMobileService.

2.22.2.7 void UpdateConnectionPriority (string *guid,* ConnectionPriorityDto *priority*)

The function sends the selected emergency priority to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
priority	The new emergency connection priority.

Implemented in WcfMobileService.

2.22.2.8 void UpdateDeviceInfo (string *guid,* MobileDeviceInformationDto *deviceInfo*)

The function sends a new mobile device status information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
deviceInfo	The new mobile device information.

Implemented in WcfMobileService.

2.22.2.9 void UpdateInstrumentList (string *guid,* List< MeasurementInstrumentDto > *instruments*)

Updates the list of supported measurement instruments to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
instruments	The list of available measurement instruments at the mobile device.

Implemented in WcfMobileService.

2.22.2.10 void UpdateLocation (string *guid,* LocationInformationDto *location*)

The function sends a new location information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
location	The new location information.

Implemented in WcfMobileService.

2.22.2.11 void UpdateMedicalInfo (string *guid,* MedicalInformationDto *medicalInfo*)

The function sends a new mobile emergency client user medical information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.

medicalInfo	New user medical information.

Implemented in WcfMobileService.

2.22.2.12 void UpdatePersonalInfo (string *guid,* PersonalInformationDto *userInfo*)

The function sends a new mobile emergency client user's information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
userInfo	The new user information.

Implemented in WcfMobileService.

2.22.2.13 void UpdateRequestType (string *guid,* EmergencyTypeDto *requestType*)

The function sends the selected emergency type to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
requestType	The selected emergency type.

Implemented in WcfMobileService.

2.22.2.14 void UploadMeasurementData (string *guid,* MeasurementInstrumentDto *instrument,* byte[] *measurementData*)

The function uploads a segment of the measurement data from the instrument at the mobile device to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
instrument	The measurement instrument.
measurement↩ Data	The measurement data in bytes.

Implemented in WcfMobileService.

2.22.2.15 void UploadMediaSegment (string *guid,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function uploads a segment of media from the mobile emergency client to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
mediaInfo	The description of the media.
mediaData	The media data bytes.

Implemented in WcfMobileService.

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

- IWcfMobileService.cs

2.23 LocationInformation Class Reference

Class for geological location information. Contains decimal format WGS84 coordinates, UTC acquisition time, accuracy in meters and a LocationType identifying the type of the stored location information.

Properties

- int Id `[get, set]`

  The function gets or sets the primary key identifying this location instance in the database.
- int ConnectionId `[get, set]`

  The function gets or sets the foreign key identifying the Connection database entity to which this location information is related.
- double Latitude `[get, set]`

  The function gets or sets WGS84 latitude in decimal format.
- double Longitude `[get, set]`

  The function gets or sets WGS84 longitude in decimal format.
- DateTimeOffset AcquisitionTime `[get, set]`

  The function gets or sets location acquisition UTC time.
- double AccuracyMeters `[get, set]`

  The function gets or sets the estimated accuracy of the location information in positive meters.
- LocationType LocationType `[get, set]`

  The function gets or sets the eype of the location information.

2.23.1 Detailed Description

Class for geological location information. Contains decimal format WGS84 coordinates, UTC acquisition time, accuracy in meters and a LocationType identifying the type of the stored location information.

<author>Veli-Mikko Puupponen</author>

2.23.2 Property Documentation

2.23.2.1 double AccuracyMeters `[get],[set]`

The function gets or sets the estimated accuracy of the location information in positive meters.

2.23.2.2 DateTimeOffset AcquisitionTime `[get],[set]`

The function gets or sets location acquisition UTC time.

2.23.2.3 int ConnectionId `[get],[set]`

The function gets or sets the foreign key identifying the Connection database entity to which this location information is related.

2.23.2.4 int Id [get], [set]

The function gets or sets the primary key identifying this location instance in the database.

2.23.2.5 double Latitude [get], [set]

The function gets or sets WGS84 latitude in decimal format.

2.23.2.6 LocationType LocationType [get], [set]

The function gets or sets the eype of the location information.

2.23.2.7 double Longitude [get], [set]

The function gets or sets WGS84 longitude in decimal format.

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

- LocationInformation.cs

2.24 LocationInformationDto Class Reference

The dto class coresponds to the LocationInformation class.

Properties

- LocationTypeDto LocationType [get, set]

  The function gets or sets the type of the location.
- double Latitude [get, set]

  The function gets or sets WGS84 latitude in decimal format.
- double Longitude [get, set]

  The function gets or sets WGS84 longitude in decimal format.
- DateTimeOffset AcquisitionTime [get, set]

  The function gets or sets location acquisition UTC time.
- double AccuracyMeters [get, set]

  The function gets or sets the estimated accuracy of the location information in positive meters.

2.24.1 Detailed Description

The dto class coresponds to the LocationInformation class.

<author>Veli-Mikko Puupponen</author> It stores geological location information. Contains decimal format WGS84 coordinates, UTC acquisition time, accuracy in meters and a LocationType identifying the type of the stored location information.

2.24.2 Property Documentation

2.24.2.1 double AccuracyMeters [get], [set]

The function gets or sets the estimated accuracy of the location information in positive meters.

2.24.2.2 DateTimeOffset AcquisitionTime [get], [set]

The function gets or sets location acquisition UTC time.

2.24.2.3 double Latitude [get], [set]

The function gets or sets WGS84 latitude in decimal format.

2.24.2.4 LocationTypeDto LocationType [get], [set]

The function gets or sets the type of the location.

2.24.2.5 double Longitude [get], [set]

The function gets or sets WGS84 longitude in decimal format.

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

- LocationInformationDto.cs

2.25 MeasurementInstrument Class Reference

Class for storing measurement instrument's identifying name, textual description, instrment type and the basic structure description of the binary data that it provides.

Properties

- MeasurementInstrumentType DeviceType [get, set]

  The function gets or sets the type of this measurement instrument.
- string DeviceIdentifier [get, set]

  The function gets or sets the textual identifier for this measurement instrument.
- string DeviceDescription [get, set]

  The function gets or sets the textual description of this measurement instrument.
- int DataHeaderLength [get, set]

  The function gets or sets the length of data header used in the binary data packets sent by this measurement instrument. Value in represent the count of octets.
- int DataSampleSize [get, set]

  The function gets or sets the length of a data sample produced by this measurement instrument. Value represents the count of octets.
- int DataSampleChannels [get, set]

  The function gets or sets the number of equivalent data channels present in the data provided by this measurement instrument.
- int DataSamplesPerSecond [get, set]

  The function gets or sets the number of data samples in second in the data provided by this measurement instrument.
- Boolean HeaderRepeatsOnEveryPacket [get, set]

  The function gets or sets the value specifying if a header of the DataHeaderLength length precedes data in every measurement data data segment provided by this measurement instrument.

2.25.1 Detailed Description

Class for storing measurement instrument's identifying name, textual description, instrment type and the basic structure description of the binary data that it provides.

<author>Veli-Mikko Puupponen</author>

2.25.2 Property Documentation

2.25.2.1 int DataHeaderLength [get],[set]

The function gets or sets the length of data header used in the binary data packets sent by this measurement instrument. Value in represent the count of octets.

2.25.2.2 int DataSampleChannels [get],[set]

The function gets or sets the number of equivalent data channels present in the data provided by this measurement instrument.

2.25.2.3 int DataSampleSize [get],[set]

The function gets or sets the length of a data sample produced by this measurement instrument. Value represents the count of octets.

2.25.2.4 int DataSamplesPerSecond [get],[set]

The function gets or sets the number of data samples in second in the data provided by this measurement instrument.

2.25.2.5 string DeviceDescription [get],[set]

The function gets or sets the textual description of this measurement instrument.

2.25.2.6 string DeviceIdentifier [get],[set]

The function gets or sets the textual identifier for this measurement instrument.

2.25.2.7 MeasurementInstrumentType DeviceType [get],[set]

The function gets or sets the type of this measurement instrument.

2.25.2.8 Boolean HeaderRepeatsOnEveryPacket [get],[set]

The function gets or sets the value specifying if a header of the DataHeaderLength length precedes data in every measurement data data segment provided by this measurement instrument.

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

- MeasurementInstrument.cs

2.26 MeasurementInstrumentDto Class Reference

The dto class corresponds to the MeasurementInstrument class.

Properties

- MeasurementInstrumentTypeDto DeviceType [get, set]

  The function gets or sets the type of this measurement instrument.
- string DeviceIdentifier [get, set]

  The function gets or sets the textual identifier for this measurement instrument.
- string DeviceDescription [get, set]

  The function gets or sets the textual description of this measurement instrument.
- int DataHeaderLength [get, set]

  The function gets or sets the length of data header used in the binary data packets sent by this measurement instrument. Value in represent the count of octets.

- int DataSampleSize [get, set]

  The function gets or sets the length of a data sample produced by this measurement instrument. Value represents the count of octets.
- int DataSampleChannels [get, set]

  The function gets or sets the number of equivalent data channels present in the data provided by this measurement instrument.
- int DataSamplesPerSecond [get, set]

  The function gets or sets the number of data samples in second in the data provided by this measurement instrument.
- Boolean HeaderRepeatsOnEveryPacket [get, set]

  The function gets or sets the value specifying whether a the DataHeaderLength length provided by this measurement instrument precedes data in every measurement data segment or not.

2.26.1 Detailed Description

The dto class corresponds to the MeasurementInstrument class.

It stores measurement instrument's identifying name, textual description, instrment type and the basic structure description of the binary data that it provides.

2.26.2 Property Documentation

2.26.2.1 int DataHeaderLength [get],[set]

The function gets or sets the length of data header used in the binary data packets sent by this measurement instrument. Value in represent the count of octets.

2.26.2.2 int DataSampleChannels [get],[set]

The function gets or sets the number of equivalent data channels present in the data provided by this measurement instrument.

2.26.2.3 int DataSampleSize [get],[set]

The function gets or sets the length of a data sample produced by this measurement instrument. Value represents the count of octets.

2.26.2.4 int DataSamplesPerSecond [get],[set]

The function gets or sets the number of data samples in second in the data provided by this measurement instrument.

2.26.2.5 string DeviceDescription [get],[set]

The function gets or sets the textual description of this measurement instrument.

2.26.2.6 string DeviceIdentifier [get],[set]

The function gets or sets the textual identifier for this measurement instrument.

2.26.2.7 MeasurementInstrumentTypeDto DeviceType [get],[set]

The function gets or sets the type of this measurement instrument.

2.26.2.8 Boolean HeaderRepeatsOnEveryPacket [get],[set]

The function gets or sets the value specifying whether a the DataHeaderLength length provided by this measurement instrument precedes data in every measurement data segment or not.

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

- MeasurementInstrumentDto.cs

2.27 MediaConfigurationDto Class Reference

The dto container class represents a audio/video capture configuration. It contains quantifiers specifying the quality of the captured media.

Public Member Functions

- void PerformConstraintCheck ()

  The function ensures that the AudioCompressionQuality and PictureCompressionQuality conform to their permissible ranges.

Public Attributes

- int AudioCompressionQualityMin = 0

  The lowest permissible value for the audio compression quality quantifier.
- int AudioCompressionQualityMax = 10

  The highest permissible value for the audio compression quality quantifier.
- const float PictureFpsMin = 0.0F

  The smallest permissible picture fps.
- int PictureCompressionQualityMin = 0

  The lowest permissible picture compression quality quantifier value.
- int PictureCompressionQualityMax = 100

  The highest permissible picture compression quality quantifier value.

Properties

- bool EnablePicture `[get, set]`

  The function gets or sets the value indicating whether image should be captured.
- bool EnableAudio `[get, set]`

  The function gets or sets the value indicating whether audio should be captured
- int AudioCompressionQuality `[get, set]`

  The function gets or sets the audio compression quality quantifier.
- float PictureFps `[get, set]`

  The function gets or sets the picture fps. Values under 1 will result in frame rates with fewer than one frames per second
- int PictureCompressionQuality `[get, set]`

  The function gets or sets the picture compression quality quantifier. This value must be in the range specified by the PictureCompressionQualityMin and PictureCompressionQualityMax.
- int PictureResolution `[get, set]`

  The function gets or sets the picture resolution quantifier value. This value should be between 0 and 10.

2.27.1 Detailed Description

The dto container class represents a audio/video capture configuration. It contains quantifiers specifying the quality of the captured media.

<author>Veli-Mikko Puupponen</author>

2.27.2 Member Function Documentation

2.27.2.1 void PerformConstraintCheck ( )

The function ensures that the AudioCompressionQuality and PictureCompressionQuality conform to their permissible ranges.

2.27.3 Member Data Documentation

2.27.3.1 int AudioCompressionQualityMax = 10

The highest permissible value for the audio compression quality quantifier.

2.27.3.2 int AudioCompressionQualityMin = 0

The lowest permissible value for the audio compression quality quantifier.

2.27.3.3 int PictureCompressionQualityMax = 100

The highest permissible picture compression quality quantifier value.

2.27.3.4 int PictureCompressionQualityMin = 0

The lowest permissible picture compression quality quantifier value.

2.27.3.5 const float PictureFpsMin = 0.0F

The smallest permissible picture fps.

2.27.4 Property Documentation

2.27.4.1 int AudioCompressionQuality [get], [set]

The function gets or sets the audio compression quality quantifier.

2.27.4.2 bool EnableAudio [get], [set]

The function gets or sets the value indicating whether audio should be captured

2.27.4.3 bool EnablePicture [get], [set]

The function gets or sets the value indicating whether image should be captured.

2.27.4.4 int PictureCompressionQuality [get], [set]

The function gets or sets the picture compression quality quantifier. This value must be in the range specified by the PictureCompressionQualityMin and PictureCompressionQualityMax.

2.27.4.5 float PictureFps [get], [set]

The function gets or sets the picture fps. Values under 1 will result in frame rates with fewer than one frames per second

2.27.4.6 int PictureResolution [get], [set]

The function gets or sets the picture resolution quantifier value. This value should be between 0 and 10.

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

- MediaConfigurationDto.cs

2.28 MediaInformationDto Class Reference

The dto container class for audio/video media description. It contains a property identifying if the payload data has been compressed with GZip for transport. Contains a MediaTypeDto instance describing the media format.

Properties

- bool CompressedGZip `[get, set]`

  The function gets or sets the value indicating if the assosicated payload data has been compressed with GZip for transfer.
- MediaTypeDto MediaType `[get, set]`

  The function gets or sets the type description of the associated payload data.

2.28.1 Detailed Description

The dto container class for audio/video media description. It contains a property identifying if the payload data has been compressed with GZip for transport. Contains a MediaTypeDto instance describing the media format.

<author>Veli-Mikko Puupponen</author>

2.28.2 Property Documentation

2.28.2.1 bool CompressedGZip `[get],[set]`

The function gets or sets the value indicating if the assosicated payload data has been compressed with GZip for transfer.

2.28.2.2 MediaTypeDto MediaType `[get],[set]`

The function gets or sets the type description of the associated payload data.

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

- MediaInformationDto.cs

2.29 MedicalInformationDto Class Reference

The dto class is for medical information.

2.29.1 Detailed Description

The dto class is for medical information.

<author>Veli-Mikko Puupponen</author> TODO: unimplemented.

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

- MedicalInformationDto.cs

2.30 MobileClientController Class Reference

The class is for interacts with an emergency client. it functions as a broker between the ConnectionController and the SignalR and WCF interfaces used by the mobile emergency client. Maintains a reference to the SignalR IHubConnectionContext required for the communication from the server to the connected mobile emergency clients.

Inherits IMobileClientMethods.

Public Member Functions

- string ClientConnected (string transportId)

  The function handles mobile emergency client connection initialization.
- void ClientReconnected (string guid, string transportId)

  The function handles mobile emergency client reconnect event.
- void ClientDisconnected (string guid)

  The function handles mobile emergency client disconnect event.
- void RequestLocationUpdate (String transportId)

  The function sends location information update request to the mobile emergency client with the specified SignalR connection ID.
- void RequestDeviceInfo (String transportId)

  The function sends mobile device status information update request to the mobile emergency client with the specified SignalR connection ID.
- void UpdateConnectionStatus (String transportId, ConnectionStateDto status)

  The function sends an updated connection status to the mobile emergency client with the specified SignalR connection ID.
- void RequestUserInfo (String transportId)

  The function sends user information update request to the mobile emergency client with the specified SignalR connection ID.
- void RequestMedicalInfo (String transportId)

  The function sends medical information update request to the mobile emergency client with the specified SignalR connection ID.
- void DisplayUserLocationMap (String transportId)

  Orders the mobile emergency client to display a map enabling the user to manually specify the current location.
- void CloseUserLocationMap (String transportId)

  Orders the mobile emergency client to hide the location map.
- void GetInstrumentList (String transportId)

  The function requests a list of available measurement instruments from the mobile emergency client with the specified SignalR connection ID.
- void ClientUpdatedLocation (string guid, LocationInformationDto location)

  The method handles mobile emergency client location updates.
- void ClientUpdatedDeviceInfo (string guid, MobileDeviceInformationDto deviceInfo)

  The method handles mobile emergency client mobile device status information updates.
- void ClientUpdatedPersonalInfo (string guid, PersonalInformationDto userIInfo)

  The method handles mobile emergency client user information updates.
- void ClientUpdatedConnectionPriority (string guid, ConnectionPriorityDto priority)

  The method handles mobile emergency client emergency connection priority updates.
- void ClientUpdatedRequestType (string guid, EmergencyTypeDto emergencyType)

  The method handles mobile emergency client emergency type updates.
- void ClientUpdatedMedicalinfo (string guid, MedicalInformationDto medicalInfo)

  The method handles mobile emergency client user's medical information updates.
- void ClientUpdatedNoSoundStatus (string guid, bool noSound)

  The method handles mobile emergency client request for an operation without sound.
- void ClientUpdatedInstrumentList (string guid, List< MeasurementInstrumentDto > instruments)

  The method handles mobile emergency client measurement instrument list update
- void ClientUploadedMediaData (string guid, MediaInformationDto mediaInfo, byte[] mediaData)

  The method handles mobile emergency client media uploads
- void ClientUploadedMeasurementData (string guid, MeasurementInstrumentDto instrument, byte[] measurementData)

  The method handles mobile emergency client measurement data uploads
- void RequestMediaUpstreaming (String transportId, MediaConfigurationDto mediaCofiguration)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration

- void RequestMediaDownstreaming (String transportId, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and displaying media from the provided location.
- void RequestStartMeasurement (String transportId, MeasurementInstrumentDto instrument)

  The function requests the mobile emergency client with the specifiedSignalR connection ID to start measuring using the specified instrument and uploading the measurement data.
- void RequestStopMeasurement (String transportId, MeasurementInstrumentDto instrument)

  The function requests the mobile emergency client with the specifiedSignalR connection ID to stop measurement with the specified instrument and to no longer upload measurement data from it.
- void ClientSentTextMessage (string guid, TextMessageDto textMessage)

  The function handles text messages incoming from mobile emergency client
- void IncomingTextMessage (string transportId, TextMessageDto textMessage)

  The function sends a text based message to the mobile emergency client with the specified SignalR connection ID.
- void RequestConnectionLatencyInfo (String transportId)

  The function requests the mobile emergency client with the specified SignalR connection ID to start connection latency measurement.
- void ClientUpdatedConnectionLatencyInfo (string guid, ConnectionLatencyInformationDto latencyInfo)

  The method handles mobile emergency client latency information update.
- AudioVideoContainerDto ClientGetMediaSegment (string guid)

  The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.

Properties

- static MobileClientController Instance `[get]`

  The function gets the MobileClientController instance.

2.30.1 Detailed Description

The class is for interacts with an emergency client. it functions as a broker between the ConnectionController and the SignalR and WCF interfaces used by the mobile emergency client. Maintains a reference to the SignalR IHubConnectionContext required for the communication from the server to the connected mobile emergency clients.

<author>Veli-Mikko Puupponen, Ilkka Rautiainen</author> This singleton is instantialized by the first call to either of the emergency client interfaces or by the ConnectionController.

See also

HalyriServer.Services.IWcfMobileService, HalyriServer.Services.WcfMobileService, HalyriServer.Services.IMobileClientMethods, HalyriServer.Services.SignalRMobileHub

2.30.2 Member Function Documentation

2.30.2.1 string ClientConnected (string *transportId*)

The function handles mobile emergency client connection initialization.

Parameters

transportId	SignalR connection id for the client connection.

Returns

GUID identifying the emergency client.

2.30.2.2 void ClientDisconnected (string *guid*)

The function handles mobile emergency client disconnect event.

Parameters

guid	GUID identifying the emergency client.

2.30.2.3 AudioVideoContainerDto ClientGetMediaSegment (string *guid*)

The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.

Parameters

guid	The guid identifying the emergency client.

Returns

A AudioVideoContainerDto containing audio/video from the call center client. If no media is available, AudioVideoContainerDto is empty with no payload.

2.30.2.4 void ClientReconnected (string *guid,* string *transportId*)

The function handles mobile emergency client reconnect event.

Parameters

guid	GUID identifying the emergency client.
transportId	SignalR connection id for the client connection.

2.30.2.5 void ClientSentTextMessage (string *guid,* TextMessageDto *textMessage*)

The function handles text messages incoming from mobile emergency client

Parameters

guid	The guid identifying the emergency client.
textMessage	The incoming text message.

2.30.2.6 void ClientUpdatedConnectionLatencyInfo (string *guid,* ConnectionLatencyInformationDto *latencyInfo*)

The method handles mobile emergency client latency information update.

Parameters

guid	The guid identifying the emergency client.
latencyInfo	The latency information.

2.30.2.7 void ClientUpdatedConnectionPriority (string *guid,* ConnectionPriorityDto *priority*)

The method handles mobile emergency client emergency connection priority updates.

Parameters

guid	The guid identifying the emergency client.
priority	The new connection priority.

2.30.2.8 void ClientUpdatedDeviceInfo (string *guid,* MobileDeviceInformationDto *deviceInfo*)

The method handles mobile emergency client mobile device status information updates.

Parameters

guid	GUID identifying the emergency client.
devideInfo	New location information.

2.30.2.9 void ClientUpdatedInstrumentList (string *guid,* List< **MeasurementInstrumentDto** > *instruments*)

The method handles mobile emergency client measurement instrument list update

Parameters

guid	The guid identifying the emergency client.
instruments	The list of available measurement instruments attached to the mobile device.

2.30.2.10 void ClientUpdatedLocation (string *guid,* **LocationInformationDto** *location*)

The method handles mobile emergency client location updates.

Parameters

guid	GUID identifying the emergency client.
location	New location information.

2.30.2.11 void ClientUpdatedMedicalInfo (string *guid,* **MedicalInformationDto** *medicalInfo*)

The method handles mobile emergency client user's medical information updates.

Parameters

guid	The guid identifying the emergency client.
medicalInfo	The new location information.

2.30.2.12 void ClientUpdatedNoSoundStatus (string *guid,* bool *noSound*)

The method handles mobile emergency client request for an operation without sound.

Parameters

guid	The guid identifying the emergency client.
noSound	True for operation without sound, otherwise False.

2.30.2.13 void ClientUpdatedPersonalInfo (string *guid,* **PersonalInformationDto** *userInfo*)

The method handles mobile emergency client user information updates.

Parameters

guid	GUID identifying the emergency client.
userInfo	New location information.

2.30.2.14 void ClientUpdatedRequestType (string *guid,* **EmergencyTypeDto** *emergencyType*)

The method handles mobile emergency client emergency type updates.

Parameters

guid	The guid identifying the emergency client.

requestType	The new location information.

2.30.2.15 void ClientUploadedMeasurementData (string *guid,* MeasurementInstrumentDto *instrument,* byte[] *measurementData*)

The method handles mobile emergency client measurement data uploads

Parameters

guid	The guid identifying the emergency client.
instrument	The measurement instrument from which the data originates.
measurement↩ Data	The measurement data.

2.30.2.16 void ClientUploadedMediaData (string *guid,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The method handles mobile emergency client media uploads

Parameters

guid	The guid identifying the emergency client.
mediaInfo	The information identifying the media type.
mediaData	The media data.

2.30.2.17 void CloseUserLocationMap (String *transportId*)

Orders the mobile emergency client to hide the location map.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.18 void DisplayUserLocationMap (String *transportId*)

Orders the mobile emergency client to display a map enabling the user to manually specify the current location.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.19 void GetInstrumentList (String *transportId*)

The function requests a list of available measurement instruments from the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.20 void IncomingTextMessage (string *transportId,* TextMessageDto *textMessage*)

The function sends a text based message to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
textMessage	The text message to be sent to the mobile emergency client.

2.30.2.21 void RequestConnectionLatencyInfo (String *transportId*)

The function requests the mobile emergency client with the specified SignalR connection ID to start connection latency measurement.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.22 void RequestDeviceInfo (String *transportId*)

The function sends mobile device status information update request to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.23 void RequestLocationUpdate (String *transportId*)

The function sends location information update request to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.24 void RequestMediaDownstreaming (String *transportId,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and displaying media from the provided location.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
mediaUrl	The location of the media to be displayed by the mobile emergency client.

Implements IMobileClientMethods.

2.30.2.25 void RequestMediaUpstreaming (String *transportId,* MediaConfigurationDto *mediaCofiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
mediaCofiguration	The configuration describing the requested media.

Implements IMobileClientMethods.

2.30.2.26 void RequestMedicalInfo (String *transportId*)

The function sends medical information update request to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.27 void RequestStartMeasurement (String *transportId,* MeasurementInstrumentDto *instrument*)

The function requests the mobile emergency client with the specifiedSignalR connection ID to start measuring using the specified instrument and uploading the measurement data.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.
instrument	Target measurement instrument.

Implements IMobileClientMethods.

2.30.2.28 void RequestStopMeasurement (String *transportId,* MeasurementInstrumentDto *instrument*)

The function requests the mobile emergency client with the specifiedSignalR connection ID to stop measurement with the specified instrument and to no longer upload measurement data from it.

Parameters

transportId	The SignalR connection ID for a mobile emergency client connection.
instrument	The target measurement instrument.

Implements IMobileClientMethods.

2.30.2.29 void RequestUserInfo (String *transportId*)

The function sends user information update request to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.

Implements IMobileClientMethods.

2.30.2.30 void UpdateConnectionStatus (String *transportId,* ConnectionStateDto *status*)

The function sends an updated connection status to the mobile emergency client with the specified SignalR connection ID.

Parameters

transportId	SignalR connection ID for a mobile emergency client connection.
status	New connection state.

Implements IMobileClientMethods.

2.30.3 Property Documentation

2.30.3.1 MobileClientController Instance [static],[get]

The function gets the MobileClientController instance.

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

- MobileClientController.cs

2.31 MobileDeviceInformation Class Reference

Class for storing status information about the mobile device running the emergency mobile client. Contains the remaining battery capacity in percents, the remaining discharge time in minutes and network capability parameters.

Properties

- int Id `[get, set]`

  The function gets or sets the primary key identifying this device information instance in the database.
- DateTime ArrivalTime `[get, set]`

  The function gets or sets the DateTime specifying the moment in time when this MobileDeviceInformation first arrived to the server.
- int ConnectionId `[get, set]`

  The function gets or sets the foreign key identifying the Connection database entity to which this device information is related.
- int RemainingChargePercent `[get, set]`

  The function gets or sets the mobile device's remaining battery capacity in percents.
- int RemainingDischargeTime `[get, set]`

  The function gets or sets the mobile device's remaining battery discharge time in minutes.
- string CellularMobileOperator `[get, set]`

  The function gets or sets the name of the cellular operator used by the mobile device.
- Boolean NetworkAvailable `[get, set]`

  The function gets or sets the value indicating whether cellular network is available.
- Boolean CellularEnabled `[get, set]`

  The function gets or sets the value indicating whether cellular data connection is available.
- Boolean RoamingEnabled `[get, set]`

  The function gets or sets the value indicating whether roaming is enabled on the connection used by the mobile device.
- Boolean WiFiEnabled `[get, set]`

  The function gets or sets the value indicating whether the mobile device is using a WiFi network for data transfer.

2.31.1 Detailed Description

Class for storing status information about the mobile device running the emergency mobile client. Contains the remaining battery capacity in percents, the remaining discharge time in minutes and network capability parameters.

<author>Veli-Mikko Puupponen</author>

2.31.2 Property Documentation

2.31.2.1 DateTime ArrivalTime `[get],[set]`

The function gets or sets the DateTime specifying the moment in time when this MobileDeviceInformation first arrived to the server.

2.31.2.2 Boolean CellularEnabled `[get],[set]`

The function gets or sets the value indicating whether cellular data connection is available.

2.31.2.3 string CellularMobileOperator `[get],[set]`

The function gets or sets the name of the cellular operator used by the mobile device.

2.31.2.4 int ConnectionId [get],[set]

The function gets or sets the foreign key identifying the Connection database entity to which this device information is related.

2.31.2.5 int Id [get],[set]

The function gets or sets the primary key identifying this device information instance in the database.

2.31.2.6 Boolean NetworkAvailable [get],[set]

The function gets or sets the value indicating whether cellular network is available.

2.31.2.7 int RemainingChargePercent [get],[set]

The function gets or sets the mobile device's remaining battery capacity in percents.

2.31.2.8 int RemainingDischargeTime [get],[set]

The function gets or sets the mobile device's remaining battery discharge time in minutes.

2.31.2.9 Boolean RoamingEnabled [get],[set]

The function gets or sets the value indicating whether roaming is enabled on the connection used by the mobile device.

2.31.2.10 Boolean WiFiEnabled [get],[set]

The function gets or sets the value indicating whether the mobile device is using a WiFi network for data transfer.

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

- MobileDeviceInformation.cs

2.32 MobileDeviceInformationDto Class Reference

The dto class corresponds to the MobileDeviceInformation class.

Properties

- int RemainingChargePercent [get, set]

  The function gets or sets the mobile device's remaining battery capacity in percents.
- int RemainingDischargeTime [get, set]

  The function gets or sets the mobile device's remaining battery discharge time in minutes.
- string CellularMobileOperator [get, set]

  The function gets or sets the name of the cellular operator used by the mobile device.
- Boolean NetworkAvailable [get, set]

  The function gets or sets the value indicating whether cellular network is available.
- Boolean CellularEnabled [get, set]

  The function gets or sets the value indicating whether cellular data connection is available.
- Boolean RoamingEnabled [get, set]

  The function gets or sets the value indicating whether roaming is enabled on the connection used by the mobile device.
- Boolean WiFiEnabled [get, set]

  The function gets or sets the value indicating whether the mobile device is using a WiFi network for data transfer.

2.32.1 Detailed Description

The dto class corresponds to the MobileDeviceInformation class.

<author>Veli-Mikko Puupponen</author> It is a container class for storing status information about the mobile device running the emergency mobile client. It contains the remaining battery capacity in percents, the remaining discharge time in minutes and network capability parameters.

2.32.2 Property Documentation

2.32.2.1 Boolean CellularEnabled [get], [set]

The function gets or sets the value indicating whether cellular data connection is available.

2.32.2.2 string CellularMobileOperator [get], [set]

The function gets or sets the name of the cellular operator used by the mobile device.

2.32.2.3 Boolean NetworkAvailable [get], [set]

The function gets or sets the value indicating whether cellular network is available.

2.32.2.4 int RemainingChargePercent [get], [set]

The function gets or sets the mobile device's remaining battery capacity in percents.

2.32.2.5 int RemainingDischargeTime [get], [set]

The function gets or sets the mobile device's remaining battery discharge time in minutes.

2.32.2.6 Boolean RoamingEnabled [get], [set]

The function gets or sets the value indicating whether roaming is enabled on the connection used by the mobile device.

2.32.2.7 Boolean WiFiEnabled [get], [set]

The function gets or sets the value indicating whether the mobile device is using a WiFi network for data transfer.

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

- MobileDeviceInformationDto.cs

2.33 ParameterFault Class Reference

The fault class is used for exceptions related to method parameters.

Inherits Fault.

Public Member Functions

- ParameterFault (String cause, string detail)

  The function initializes a new ParameterFault instance with the specified cause description and a detailed fault description.

- ParameterFault (String cause)

  The function initializes a new ParameterFault instance with the specified cause description.

Additional Inherited Members

2.33.1 Detailed Description

The fault class is used for exceptions related to method parameters.

<author>Veli-Mikko Puupponen</author>

2.33.2 Constructor & Destructor Documentation

2.33.2.1 ParameterFault (String *cause*, string *detail*)

The function initializes a new ParameterFault instance with the specified cause description and a detailed fault description.

Parameters

cause	The dscription of the cause.
detail	The more detailed description of the exception.

2.33.2.2 ParameterFault (String *cause*)

The function initializes a new ParameterFault instance with the specified cause description.

Parameters

cause	The description of the cause.

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

- ParameterFault.cs

2.34 PersonalInformation Class Reference

Class for storing emergency mobile client user's personal information. Contains user name, street address, postal code, locality and a list of user's telephone numbers.

Properties

- int Id `[get, set]`

  The function gets or sets the primary key identifying this personal information instance in the database.
- int ConnectionId `[get, set]`

  The function gets or sets the foreign key identifying the Connection database entity to which this personal information is related.
- string Name `[get, set]`

  The function gets or sets the name of the user.
- string StreetAddress `[get, set]`

  The function gets or sets the street address of the user.
- int PostalCode `[get, set]`

  The function gets or sets the postal code of the user's address.
- string Locality `[get, set]`

  The function gets or sets the locality of the user's address.
- List< string > PhoneNumbers `[get, set]`

  The function gets or sets the list of the user's telephone numbers.

2.34.1 Detailed Description

Class for storing emergency mobile client user's personal information. Contains user name, street address, postal code, locality and a list of user's telephone numbers.

<author>Veli-Mikko Puupponen</author>

2.34.2 Property Documentation

2.34.2.1 int ConnectionId [get],[set]

The function gets or sets the foreign key identifying the Connection database entity to which this personal information is related.

2.34.2.2 int Id [get],[set]

The function gets or sets the primary key identifying this personal information instance in the database.

2.34.2.3 string Locality [get],[set]

The function gets or sets the locality of the user's address.

2.34.2.4 string Name [get],[set]

The function gets or sets the name of the user.

2.34.2.5 List<string> PhoneNumbers [get],[set]

The function gets or sets the list of the user's telephone numbers.

2.34.2.6 int PostalCode [get],[set]

The function gets or sets the postal code of the user's address.

2.34.2.7 string StreetAddress [get],[set]

The function gets or sets the street address of the user.

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

- PersonalInformation.cs

2.35 PersonalInformationDto Class Reference

The dto class corresponds to the PersonalInformation class.

Properties

- string Name [get, set]
  The function gets or sets the name of the user.
- string StreetAddress [get, set]
  The function gets or sets the street address of the user.
- int PostalCode [get, set]
  The function gets or sets the postal code of the user's address.
- string Locality [get, set]
  The function gets or sets the locality of the user's address.
- List< string > PhoneNumbers [get, set]
  The function gets or sets the list of the user's telephone numbers.

2.35.1 Detailed Description

The dto class corresponds to the PersonalInformation class.

<author>Veli-Mikko Puupponen</author> It is a container class for storing emergency mobile client user's personal information. It contains user name, street address, postal code, locality and a list of user's telephone numbers.

2.35.2 Property Documentation

2.35.2.1 string Locality [get],[set]

The function gets or sets the locality of the user's address.

2.35.2.2 string Name [get],[set]

The function gets or sets the name of the user.

2.35.2.3 List<string> PhoneNumbers [get],[set]

The function gets or sets the list of the user's telephone numbers.

2.35.2.4 int PostalCode [get],[set]

The function gets or sets the postal code of the user's address.

2.35.2.5 string StreetAddress [get],[set]

The function gets or sets the street address of the user.

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

- PersonalInformationDto.cs

2.36 SignalRMobileHub Class Reference

SignalR hub for the mobile emergency client. It supports emergency client connect, disconnect and reconnect methods. Functions as the callback channel for server-to-client asynchronous method invocations.

Inherits Hub.

Public Member Functions

- SignalRMobileHub ()

  Instantializes the SignalR hub for mobile emergency client. It gets the MobileClientController instance.
- string Connect ()

  The function opens a new emergency connection in the system for the mobile emergency client. A GUID is assigned and returned to the client. All subsequest method invocations during the open connection will use the supplied GUID.
- void Reconnect (string guid)

  The method for explicitly reconnecting mobile mergency client to the system.
- override Task OnReconnected ()

  The function handles mobile client reconnect events. It updates the SignalR connection ID used for callbacks from server to the mobile emergency client.
- void Disconnect (string guid)

  The function closes the emergency connection. Should be called by all mobile emergency clients after their connection has been signaled as handled.
- override Task OnDisconnected ()

  The function handles mobile client disconnect events.

2.36.1 Detailed Description

SignalR hub for the mobile emergency client. It supports emergency client connect, disconnect and reconnect methods. Functions as the callback channel for server-to-client asynchronous method invocations.

<author>Veli-Mikko Puupponen</author>

2.36.2 Constructor & Destructor Documentation

2.36.2.1 SignalRMobileHub ( )

Instantializes the SignalR hub for mobile emergency client. It gets the MobileClientController instance.

2.36.3 Member Function Documentation

2.36.3.1 string Connect ( )

The function opens a new emergency connection in the system for the mobile emergency client. A GUID is assigned and returned to the client. All subsequest method invocations during the open connection will use the supplied G↩ UID.

This is the first method to be called by all connecting mobile emergency clients.

Returns

The guid for the connection being opened.

2.36.3.2 void Disconnect (string *guid*)

The function closes the emergency connection. Should be called by all mobile emergency clients after their connection has been signaled as handled.

Parameters

guid	The guid identifying the mobile emergency client.

2.36.3.3 override Task OnDisconnected ( )

The function handles mobile client disconnect events.

Returns

Returns an asynchronous taks to trigger disconnect for the client.

2.36.3.4 override Task OnReconnected ( )

The function handles mobile client reconnect events. It updates the SignalR connection ID used for callbacks from server to the mobile emergency client.

Returns

Returns an asynchronous taks to trigger reconnection for the client.

2.36.3.5 void Reconnect (string *guid*)

The method for explicitly reconnecting mobile mergency client to the system.

Parameters

guid	The existing GUID for the reconnecting mobile emergency client.

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

- SignalRMobileHub.cs

2.37 Startup Class Reference

This class is mostly generated code to start the application. It also starts SignalR.

Public Member Functions

- void Configuration (IAppBuilder app)

  The function to configure and the app to be started.

2.37.1 Detailed Description

This class is mostly generated code to start the application. It also starts SignalR.

2.37.2 Member Function Documentation

2.37.2.1 void Configuration (IAppBuilder *app*)

The function to configure and the app to be started.

Parameters

app	The current app.

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

- Startup.cs

2.38 TargetStateFault Class Reference

The fault class is used for exceptions related to target instance state exceptions.

Inherits Fault.

Public Member Functions

- TargetStateFault (String cause, string detail)

  The function initializes a new TargetStateFault instance with the specified cause description and a detailed fault description.
- TargetStateFault (String cause)

  The function initializes a new TargetStateFault instance with the specified cause description.

Additional Inherited Members

2.38.1 Detailed Description

The fault class is used for exceptions related to target instance state exceptions.

<author>Veli-Mikko Puupponen</author>

2.38.2 Constructor & Destructor Documentation

2.38.2.1 TargetStateFault (String *cause,* string *detail*)

The function initializes a new TargetStateFault instance with the specified cause description and a detailed fault description.

Parameters

cause	The description of the cause.
detail	The more detailed description of the exception.

2.38.2.2 TargetStateFault (String *cause*)

The function initializes a new TargetStateFault instance with the specified cause description.

Parameters

cause	The description of the cause.

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

- TargetStateFault.cs

2.39 TextMessage Class Reference

The class is used for text based short messages. It stores the message content, the message originator and the server timestamp of the message.

Public Types

- enum MessageOriginator { MobileClient, CallCenterClient }

  The message originator The enumeration differentiating MobileClient and CallCenterClient.

Properties

- DateTime TimeStamp `[get, set]`

  The function gets or sets the DateTime specifying the moment in time when this TextMessage first arrived to the server.
- string Content `[get, set]`

  The function gets or sets the message content.
- MessageOriginator Originator `[get, set]`

  The function gets or sets the originator of this message.

2.39.1 Detailed Description

The class is used for text based short messages. It stores the message content, the message originator and the server timestamp of the message.

<author>Veli-Mikko Puupponen</author>

2.39.2 Member Enumeration Documentation

2.39.2.1 enum MessageOriginator

The message originator The enumeration differentiating MobileClient and CallCenterClient.

Enumerator

MobileClient Representing a mobile emergency client.

CallCenterClient Representing a call center client.

2.39.3 Property Documentation

2.39.3.1 string Content [get],[set]

The function gets or sets the message content.

2.39.3.2 MessageOriginator Originator [get],[set]

The function gets or sets the originator of this message.

2.39.3.3 DateTime TimeStamp [get],[set]

The function gets or sets the DateTime specifying the moment in time when this TextMessage first arrived to the server.

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

- TextMessage.cs

2.40 TextMessageDto Class Reference

The dto class corresnpods to the TextMessage class.

Public Types

- enum MessageOriginatorDto { MobileClient, CallCenterClient }

  The message originator The enumeration differentiating MobileClient and CallCenterClient.

Properties

- DateTime TimeStamp [get, set]

  The function gets or sets the DateTime specifying the moment in time when this TextMessage first arrived to the server.
- string Content [get, set]

  The function gets or sets the message content.
- MessageOriginatorDto Originator [get, set]

  The function gets or sets the originator of this message.

2.40.1 Detailed Description

The dto class corresnpods to the TextMessage class.

<author>Veli-Mikko Puupponen</author> It is a container class for text based short messages. It stores message content, message originator and message server time stamp.

2.40.2 Member Enumeration Documentation

2.40.2.1 enum MessageOriginatorDto

The message originator The enumeration differentiating MobileClient and CallCenterClient.

Enumerator

MobileClient Representing a mobile emergency client.

CallCenterClient Representing a call center client.

2.40.3 Property Documentation

2.40.3.1 string Content [get],[set]

The function gets or sets the message content.

2.40.3.2 MessageOriginatorDto Originator [get],[set]

The function gets or sets the originator of this message.

2.40.3.3 DateTime TimeStamp [get],[set]

The function gets or sets the DateTime specifying the moment in time when this TextMessage first arrived to the server.

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

- TextMessageDto.cs

2.41 UdpMediaRelayServerCore Class Reference

The media broker service core is used with the UdpMediaClientSocket instances. It implements ping reply and simple routing mechanism. Accepts routing configuration packets (ControlPacket). For every guid, there can be a single target guid to which all MediaHeaderPackets and MediaContinuationPackets from that guid are sent. If there is no valid target mapping for the guid specified in the received media packet, the packet is silently ignored.

Public Member Functions

- UdpMediaRelayServerCore (int port)

  The function initializes a new UdpMediaRelayServerCore that uses the specified port number at the local machine to provide media broker services over UDP.
- void Enable ()

  The function enables this UdpMediaRelayServerCore to accept routing packets, to reply to client ping packets and to perform media packet receiving and routing.
- void Disable ()

  The function to disable the UdpMediaRelayServerCore and releases the underlying UDP socket. After disabling, the instance will no longer receive or send any packets.
- void EnableRouting (string sourceGuid, string targetGuid)

  The function enables routing of received media packets that have the sourceGuid as their sender Guid to the client that uses the targetGuid as sender Guid in its ping packets.
- void DisableRouting (string sourceGuid, string targetGuid)

  The function disables routing of received media packets which have the sourceGuid as their sender Guid to the client that uses the targetGuid as sender Guid in its ping packets.

Public Attributes

- StatusMessage StatusMessageEvent

  The event for loggable status message events. Published when configuration or status is updated.

2.41.1 Detailed Description

The media broker service core is used with the UdpMediaClientSocket instances. It implements ping reply and simple routing mechanism. Accepts routing configuration packets (ControlPacket). For every guid, there can be a single target guid to which all MediaHeaderPackets and MediaContinuationPackets from that guid are sent. If there is no valid target mapping for the guid specified in the received media packet, the packet is silently ignored.

<author>Veli-Mikko Puupponen</author> The server performs no packet content checking for any media packet. No rules are enforced on the received routing requests. Any client can request routing from and to any other client.

The client routings can also be updated with methods EnableRouting and DisableRouting.

The client address to Guid mappings are kept up-to-date from the ping packets.

2.41.2 Constructor & Destructor Documentation

2.41.2.1 UdpMediaRelayServerCore (int *port*)

The function initializes a new UdpMediaRelayServerCore that uses the specified port number at the local machine to provide media broker services over UDP.

Parameters

port	Service's UDP port number.

2.41.3 Member Function Documentation

2.41.3.1 void Disable ( )

The function to disable the UdpMediaRelayServerCore and releases the underlying UDP socket. After disabling, the instance will no longer receive or send any packets.

2.41.3.2 void DisableRouting (string *sourceGuid,* string *targetGuid*)

The function disables routing of received media packets which have the sourceGuid as their sender Guid to the client that uses the targetGuid as sender Guid in its ping packets.

Parameters

sourceGuid	The guid of the sender.
targetGuid	The guid of the user to whom the media packets with the sourceGuid as a sender Guid are no longer sent.

2.41.3.3 void Enable ( )

The function enables this UdpMediaRelayServerCore to accept routing packets, to reply to client ping packets and to perform media packet receiving and routing.

2.41.3.4 void EnableRouting (string *sourceGuid,* string *targetGuid*)

The function enables routing of received media packets that have the sourceGuid as their sender Guid to the client that uses the targetGuid as sender Guid in its ping packets.

Parameters

sourceGuid	The guid of the sender.
targetGuid	The guid of the user to whom the media packets with the sourceGuid as a sender Guid are sent.

2.41.4 Member Data Documentation

2.41.4.1 StatusMessage StatusMessageEvent

The event for loggable status message events. Published when configuration or status is updated.

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

- UdpMediaRelayServerCore.cs

2.42 UserCredentialsDto Class Reference

The dto container class is used for user login credentials. It contains a user name and a password.

Properties

- string UserName `[get, set]`

  The function gets or sets the user name.
- string Password `[get, set]`

  The function gets or sets the user's password.

2.42.1 Detailed Description

The dto container class is used for user login credentials. It contains a user name and a password.

2.42.2 Property Documentation

2.42.2.1 string Password `[get],[set]`

The function gets or sets the user's password.

2.42.2.2 string UserName `[get],[set]`

The function gets or sets the user name.

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

- UserCredentialsDto.cs

2.43 WcfCallCenterService Class Reference

The WCF service for call center clients. It contains function interface for call center clients. It implements IWcfCall↩CenterService.

See also

HalyriServer.Services.IWcfCallCenterService

Inherits IWcfCallCenterService.

Public Member Functions

- CallCenterConnectionDto Connect (UserCredentialsDto credentials)

  The method handles call center client connections. This method has to be invoked by all call center clients before any operations on the server can be performed.
- void Reconnect (CallCenterConnectionDto user)

The method handles call center client reconnections.

- void Disconnect (CallCenterConnectionDto user)

  The method handles call center client disconnects. Every call center client should invoke this method before closing connection.

- List< ConnectionDto > GetActiveConnections (CallCenterConnectionDto user)

  The function requests a complete list of all mobile emergency client connections currently active on the server.

- void OpenConnectionForProcessing (CallCenterConnectionDto user, ConnectionDto connection)

  The function opens a mobile emergency client connection for handling in this call center connection.

- void TransferConnection (CallCenterConnectionDto user, List< CallCenterConnectionDto > targetCall↩ CenterConnections)

  The function transfers a mobile emergency client connection from the requesting call center connection to the specified call center connection(s).

- void SetConnectionPriority (CallCenterConnectionDto user, ConnectionDto connection)

  The function changes the priority for the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

- void MoveConnectionToHold (CallCenterConnectionDto user, ConnectionDto connection)

  Puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.

- void MarkProcessedCloseConnection (CallCenterConnectionDto user, ConnectionDto connection)

  The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

- void RequestRemoteAction (CallCenterConnectionDto user, ConnectionDto connection, RemoteActionDto action)

  The function requests an operation with no parameters to be executed by the provided mobile emergency connection.

- void RequestMediaUpstreaming (CallCenterConnectionDto user, ConnectionDto connection, Media↩ ConfigurationDto mediaConfiguration)

  The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

- void RequestMediaDownstreaming (CallCenterConnectionDto user, ConnectionDto connection, string mediaUrl)

  The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

- void RequestStartMeasurement (CallCenterConnectionDto user, ConnectionDto connection, Measurement↩ InstrumentDto measurementDevice)

  The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

- void RequestStopMeasurement (CallCenterConnectionDto user, ConnectionDto connection, Measurement↩ InstrumentDto measurementDevice)

  The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

- void SendTextMessage (CallCenterConnectionDto user, ConnectionDto connection, TextMessageDto text↩ Message)

  v The function sends a text based message to the specified mobile emergency client.

- bool UploadMediaSegment (CallCenterConnectionDto user, MediaInformationDto mediaInfo, byte[] media↩ Data)

  The function uploads a segment of media from the call center client to the server. The media is forwarded to the mobile emergency client attached to the connection that the call center client currently is handling, if any.

- int Ping (int pingSequence)

  The function to ping the client. It is used to determine whether the client is still connected or not.

2.43.1 Detailed Description

The WCF service for call center clients. It contains function interface for call center clients. It implements IWcfCall↩ CenterService.

See also

HalyriServer.Services.IWcfCallCenterService

<author>Veli-Mikko Puupponen</author>

2.43.2 Member Function Documentation

2.43.2.1 CallCenterConnectionDto Connect (UserCredentialsDto *credentials*)

The method handles call center client connections. This method has to be invoked by all call center clients before any operations on the server can be performed.

Parameters

credentials	The call center client login credentials.

Returns

The CallCenterConnectionDto used to identify the call center client in subsequent operations.

Implements IWcfCallCenterService.

2.43.2.2 void Disconnect (CallCenterConnectionDto *user*)

The method handles call center client disconnects. Every call center client should invoke this method before closing connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Implements IWcfCallCenterService.

2.43.2.3 List<ConnectionDto> GetActiveConnections (CallCenterConnectionDto *user*)

The function requests a complete list of all mobile emergency client connections currently active on the server.

It Throws ConnectionFault if the provided CallCenterConnection is not a valid active connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Returns

A list of active emergency connections.

Implements IWcfCallCenterService.

2.43.2.4 void MarkProcessedCloseConnection (CallCenterConnectionDto *user*, ConnectionDto *connection*)

The function closes the emergency connection and marks it processed. If there are multiple attached handlers on the connection, it is sufficient for one handler to close the connection. Change to the connection state will be propagated to other attached handlers.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.

Implements IWcfCallCenterService.

2.43.2.5 void MoveConnectionToHold (CallCenterConnectionDto *user,* ConnectionDto *connection*)

Puts an emergency connection on hold. The invocating call center connection will no longer be an attached handler of the connection.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the mobile emergency client.

Implements IWcfCallCenterService.

2.43.2.6 void OpenConnectionForProcessing (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function opens a mobile emergency client connection for handling in this call center connection.

It Throws ConnectionFault, if the provided CallCenterConnectionDto is invalid It Throws TargetStateFault if connection is already processed, does not exist or the requesting call center client is already an associated handler for this connection.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto for the connection to be opened for handling.

Implements IWcfCallCenterService.

2.43.2.7 int Ping (int *pingSequence*)

The function to ping the client. It is used to determine whether the client is still connected or not.

Parameters

pingSequence	The sequence number of the ping.

Returns

A new ping with a new sequence number.

Implements IWcfCallCenterService.

2.43.2.8 void Reconnect (CallCenterConnectionDto *user*)

The method handles call center client reconnections.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.

Implements IWcfCallCenterService.

2.43.2.9 void RequestMediaDownstreaming (CallCenterConnectionDto *user,* ConnectionDto *connection,* string *mediaUrl*)

The function requests the mobile emergency client to start downstreaming and playback of media available at the provided url.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
mediaUrl	The location of the media to be played back at the mobile client.

Implements IWcfCallCenterService.

2.43.2.10 void RequestMediaUpstreaming (CallCenterConnectionDto *user*, ConnectionDto *connection*, MediaConfigurationDto *mediaConfiguration*)

The function requests the mobile emergency client to start upstreaming media according to the provided configuration.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
media↩ Configuration	The quality parameters for the requested media.

Implements IWcfCallCenterService.

2.43.2.11 void RequestRemoteAction (CallCenterConnectionDto *user*, ConnectionDto *connection*, RemoteActionDto *action*)

The function requests an operation with no parameters to be executed by the provided mobile emergency connection.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
action	The request to be executed on the remote call center client.

Implements IWcfCallCenterService.

2.43.2.12 void RequestStartMeasurement (CallCenterConnectionDto *user*, ConnectionDto *connection*, MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to enable the specified masurement instrument and start uploading measurement data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device To be enabled and started uploading data from.

Implements IWcfCallCenterService.

2.43.2.13 void RequestStopMeasurement (CallCenterConnectionDto *user,* ConnectionDto *connection,* MeasurementInstrumentDto *measurementDevice*)

The function requests the mobile emergency client to disable the specified measurement instrument and stop uploading data from it.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
measurement↩ Device	The measurement device to disable and stop uploading data from.

Implements IWcfCallCenterService.

2.43.2.14 void SendTextMessage (CallCenterConnectionDto *user,* ConnectionDto *connection,* TextMessageDto *textMessage*)

v The function sends a text based message to the specified mobile emergency client.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
connection	The ConnectionDto identifying the target mobile emergency client connection.
textMessage	The test message to be sent to the specified emergency client.

Implements IWcfCallCenterService.

2.43.2.15 void SetConnectionPriority (CallCenterConnectionDto *user,* ConnectionDto *connection*)

The function changes the priority for the provided emergency connection. The new priority must be set on the ConnectionDto supplied as a parameter.

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection thas is a handler for the connection to be modified.

.

Parameters

connection	The ConnectionDto identifying the mobile emergency client.

Implements IWcfCallCenterService.

2.43.2.16 void TransferConnection (CallCenterConnectionDto *user,* List< CallCenterConnectionDto > *targetCallCenterConnections*)

The function transfers a mobile emergency client connection from the requesting call center connection to the specified call center connection(s).

It Throws ConnectionFault, if the CallCenterConnectionDto is not associated with a valid call center client connection
It Throws TargetStateFault if connection is already processed, does not exist or has disconnected

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection thas is a handler for the connection to be transferred.

.

Parameters

targetCall↩ Center↩ Connections	The CallCenterConnectionDtos identifying the transfer targets.

Implements IWcfCallCenterService.

2.43.2.17 bool UploadMediaSegment (CallCenterConnectionDto *user,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function uploads a segment of media from the call center client to the server. The media is forwarded to the mobile emergency client attached to the connection that the call center client currently is handling, if any.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws TargetStateFault if the connection is processed or not a connection that the invoking call center connection is processing.

Parameters

user	The CallCenterConnectionDto identifying the existing call center client user connection.
mediaInfo	The description of the media.
mediaData	The media data bytes.

Implements IWcfCallCenterService.

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

- WcfCallCenterService.svc.cs

2.44 WcfMobileService Class Reference

The WCF service for the emergency mobile clients. The service is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.

Inherits IWcfMobileService.

Public Member Functions

- void UpdateLocation (string guid, LocationInformationDto location)

  The function sends a new location information to the server.
- void UpdateDeviceInfo (string guid, MobileDeviceInformationDto deviceInfo)

  The function sends a new mobile device status information to the server.
- void UpdatePersonalInfo (string guid, PersonalInformationDto userInfo)

  The function sends a new mobile emergency client user's information to the server.
- void UpdateMedicalInfo (string guid, MedicalInformationDto medicalInfo)

  The function sends a new mobile emergency client user medical information to the server.
- void UpdateConnectionPriority (string guid, ConnectionPriorityDto priority)

  The function sends the selected emergency priority to the server.
- void UpdateRequestType (string guid, EmergencyTypeDto requestType)

  The function sends the selected emergency type to the server.
- void ToggleNoSound (string guid, bool noSound)

  The function sets mobile emergency client request for an operation without sound.
- void UpdateInstrumentList (string guid, List< MeasurementInstrumentDto > instruments)

Updates the list of supported measurement instruments to the server.

- void UploadMediaSegment (string guid, MediaInformationDto mediaInfo, byte[] mediaData)

  The function uploads a segment of media from the mobile emergency client to the server.
- void UploadMeasurementData (string guid, MeasurementInstrumentDto instrument, byte[] measurement↩ Data)

  The function uploads a segment of the measurement data from the instrument at the mobile device to the server.
- void SendTextMessage (string guid, TextMessageDto textMessage)

  The function sends a text based message to the call center client handling the emergency connection.
- byte[] SendTestPacket (string guid, byte[] testPacket)

  The function sends a test packet. If the received packet is small, a big random packet is returned. If the received packet is big, a small random packet is returned.
- void UpdateConnectionLatencyInfo (string guid, ConnectionLatencyInformationDto latencyInfo)

  Updates the connection latency information to the server.
- AudioVideoContainerDto GetMediaSegment (string guid)

  The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.
- int Ping (int pingSequence)

  The function to ping the client. It is used to determine whether the client is still connected or not.

2.44.1 Detailed Description

The WCF service for the emergency mobile clients. The service is only used for client-to-server invocation after the emergency mobile client has opened an emergency connection using the SignalR hub connection.

<author>Veli-Mikko Puupponen</author> It implements IWcfMobileService.

See also

HalyriServer.Services.IWcfMobileService

2.44.2 Member Function Documentation

2.44.2.1 AudioVideoContainerDto GetMediaSegment (string *guid*)

The function handles new audio/video media segment requests from mobile emergency clients. It returns a new audio/video media segment if one has been uploaded by a call center client handling this emergency connection. Otherwise returns an empty AudioVideoContainerDto with no media payload.

Parameters

guid	The guid identifying the emergency client.

Returns

A AudioVideoContainerDto containing audio/video from the call center client. If no media is available, Audio↩ VideoContainerDto is empty with no payload.

Implements IWcfMobileService.

2.44.2.2 int Ping (int *pingSequence*)

The function to ping the client. It is used to determine whether the client is still connected or not.

Parameters

pingSequence	The sequence number of the ping.</p

Implements IWcfMobileService.

2.44.2.3 byte [ ] SendTestPacket (string *guid,* byte[ ] *testPacket*)

The function sends a test packet. If the received packet is small, a big random packet is returned. If the received packet is big, a small random packet is returned.

It Throws ParameterFault if guid is null It Throws Fault if the test packet length is wrong

Parameters

guid	The gúid identifying the mobile emergency client connection.
testPacket	The test packet.

Returns

A big or small test packet.

Implements IWcfMobileService.

2.44.2.4 void SendTextMessage (string *guid,* TextMessageDto *textMessage*)

The function sends a text based message to the call center client handling the emergency connection.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
textMessage	The text message to the call center.

Implements IWcfMobileService.

2.44.2.5 void ToggleNoSound (string *guid,* bool *noSound*)

The function sets mobile emergency client request for an operation without sound.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
noSound	True if the operation without sound is requested, otherwise False.

Implements IWcfMobileService.

2.44.2.6 void UpdateConnectionLatencyInfo (string *guid,* ConnectionLatencyInformationDto *latencyInfo*)

Updates the connection latency information to the server.

It Throws ParameterFault if guid or latency info is null

Parameters

guid	The guid identifying the mobile emergency client connection.
latencyInfo	The latency info.

Implements IWcfMobileService.

2.44.2.7 void UpdateConnectionPriority (string *guid*, ConnectionPriorityDto *priority*)

The function sends the selected emergency priority to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
priority	The new emergency connection priority.

Implements IWcfMobileService.

2.44.2.8 void UpdateDeviceInfo (string *guid*, MobileDeviceInformationDto *deviceInfo*)

The function sends a new mobile device status information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
deviceInfo	The new mobile device information.

Implements IWcfMobileService.

2.44.2.9 void UpdateInstrumentList (string *guid*, List< MeasurementInstrumentDto > *instruments*)

Updates the list of supported measurement instruments to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
instruments	The list of available measurement instruments at the mobile device.

Implements IWcfMobileService.

2.44.2.10 void UpdateLocation (string *guid*, LocationInformationDto *location*)

The function sends a new location information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
location	The new location information.

Implements IWcfMobileService.

2.44.2.11 void UpdateMedicalInfo (string *guid,* MedicalInformationDto *medicalInfo*)

The function sends a new mobile emergency client user medical information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
medicalInfo	New user medical information.

Implements IWcfMobileService.

2.44.2.12 void UpdatePersonalInfo (string *guid,* PersonalInformationDto *userInfo*)

The function sends a new mobile emergency client user's information to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
userInfo	The new user information.

Implements IWcfMobileService.

2.44.2.13 void UpdateRequestType (string *guid,* EmergencyTypeDto *requestType*)

The function sends the selected emergency type to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
requestType	The selected emergency type.

Implements IWcfMobileService.

2.44.2.14 void UploadMeasurementData (string *guid,* MeasurementInstrumentDto *instrument,* byte[] *measurementData*)

The function uploads a segment of the measurement data from the instrument at the mobile device to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
instrument	The measurement instrument.
measurement↩ Data	The measurement data in bytes.

Implements IWcfMobileService.

2.44.2.15 void UploadMediaSegment (string *guid,* MediaInformationDto *mediaInfo,* byte[] *mediaData*)

The function uploads a segment of media from the mobile emergency client to the server.

It Throws ParameterFault if the supplied parameters are null or incorrect. It Throws ConnectionFault if the supplied GUID does not represent a valid connection. It Throws TargetStateFault if the connection is processed or otherwise in an incompatible state.

Parameters

guid	The guid identifying the mobile emergency client connection.
mediaInfo	The description of the media.
mediaData	The media data bytes.

Implements IWcfMobileService.

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

- WcfMobileService.svc.cs