pyTSon

API Documentation

November 9, 2016

Contents

1 Module plugin

1.1 Class ts3plugin

object
└ **plugin.ts3plugin**

1.1.1 Methods

__init__(*self*)

x.__init__(...) initializes x; see help(type(x)) for signature

Overrides: object.__init__

stop(*self*)

configure(*self*, *qParentWidget*)

Parameters

`qParentWidget:` *(type=)*

infoData(*self*, *schid*, *id*, *atype*)

Parameters

`schid:` *(type=)*

`id:` *(type=)*

`atype:` *(type=)*

processCommand(*self*, *schid*, *command*)

Parameters

`schid:` *(type=)*

`command:` *(type=)*

onServerErrorEvent(*self*, *schid*, *errorMessage*, *error*, *returnCode*, *extraMessage*)

Parameters

`schid:` *(type=)*

`errorMessage:` *(type=)*

`error:` *(type=)*

`returnCode:` *(type=)*

`extraMessage:` *(type=)*

onTextMessageEvent(*self*, *schid*, *targetMode*, *toID*, *fromID*, *fromName*, *fromUniqueIdentifier*, *message*, *ffIgnored*)

Parameters

`schid:`	*(type=)*
`targetMode:`	*(type=)*
`toID:`	*(type=)*
`fromID:`	*(type=)*
`fromName:`	*(type=)*
`fromUniqueIdentifier:`	*(type=)*
`message:`	*(type=)*
`ffIgnored:`	*(type=)*

onClientPokeEvent(*self*, *schid*, *fromClientID*, *pokerName*, *pokerUniqueIdentity*, *message*, *ffIgnored*)

Parameters

`schid:`	*(type=)*
`fromClientID:`	*(type=)*
`pokerName:`	*(type=)*
`pokerUniqueIdentity:`	*(type=)*
`message:`	*(type=)*
`ffIgnored:`	*(type=)*

onServerPermissionErrorEvent(*self*, *schid*, *errorMessage*, *error*, *returnCode*, *failedPermissionID*)

Parameters

`schid:`	*(type=)*
`errorMessage:`	*(type=)*
`error:`	*(type=)*
`returnCode:`	*(type=)*
`failedPermissionID:`	*(type=)*

onUserLoggingMessageEvent(*self*, *logMessage*, *logLevel*, *logChannel*, *logID*, *logTime*, *completeLogString*)

Parameters

`logMessage:`	*(type=)*
`logLevel:`	*(type=)*
`logChannel:`	*(type=)*
`logID:`	*(type=)*
`logTime:`	*(type=)*
`completeLogString:`	*(type=)*

onEditPlaybackVoiceDataEvent(*self*, *schid*, *clientID*, *samples*, *channels*)

Parameters

`schid:` *(type=)*

`samples:` *(type=)*

`channels:` *(type=)*

Return Value

(type=tuple(bool, list(int)))

onEditPostProcessVoiceDataEvent(*schid*, *clientID*, *samples*, *channels*, *channelSpeakerArray*, *channelFillMask*)

Parameters

`schid:` *(type=)*

`clientID:` *(type=)*

`samples:` *(type=)*

`channels:` *(type=)*

`channelSpeakerArray:` *(type=)*

`channelFillMask:` *(type=)*

Return Value

(type=tuple(bool, list(int), int))

onEditMixedPlaybackVoiceDataEvent(*schid*, *samples*, *channels*, *channelSpeakerArray*, *channelFillMask*)

Parameters

`schid:` *(type=)*

`samples:` *(type=)*

`channels:` *(type=)*

`channelSpeakerArray:` *(type=)*

`channelFillMask:` *(type=)*

Return Value

(type=tuple(bool, list(int), int))

onEditCapturedVoiceDataEvent(*schid*, *samples*, *channels*, *edited*)

Parameters

`schid:` *(type=)*

`samples:` *(type=)*

`channels:` *(type=)*

`edited:` *(type=)*

Return Value

(type=tuple(bool, list(int), int))

onCustom3dRolloffCalculationClientEvent(*schid*, *clientID*, *distance*, *volume*)

Parameters

`schid:` *(type=)*

`clientID:` *(type=)*

`distance:` *(type=)*

`volume:` *(type=)*

Return Value

(type=float)

onCustom3dRolloffCalculationWaveEvent(*schid*, *waveHandle*, *distance*, *volume*)

Parameters

`schid:` *(type=)*

`waveHandle:` *(type=)*

`distance:` *(type=)*

`volume:` *(type=)*

Return Value

(type=float)

onServerStopEvent(*self*, *serverConnectionHandlerID*, *shutdownMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`shutdownMessage:` *(type=)*

onClientDBIDfromUIDEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientDatabaseID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`uniqueClientIdentifier:` *(type=)*

`clientDatabaseID:` *(type=)*

onBanListEvent(*self*, *serverConnectionHandlerID*, *banid*, *ip*, *name*, *uid*, *creationTime*, *durationTime*, *invokerName*, *invokercldbid*, *invokeruid*, *reason*, *numberOfEnforcements*, *lastNickName*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`banid:`	*(type=)*
`ip:`	*(type=)*
`name:`	*(type=)*
`uid:`	*(type=)*
`creationTime:`	*(type=)*
`durationTime:`	*(type=)*
`invokerName:`	*(type=)*
`invokercldbid:`	*(type=)*
`invokeruid:`	*(type=)*
`reason:`	*(type=)*
`numberOfEnforcements:`	*(type=)*
`lastNickName:`	*(type=)*

currentServerConnectionChanged(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:`	id of the new serverconnectionhandler *(type=long)*

onServerConnectionInfoEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*

onPlaybackShutdownCompleteEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*

onDelChannelEvent(*self*, *serverConnectionHandlerID*, *channelID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`invokerID:`	*(type=)*
`invokerName:`	*(type=)*
`invokerUniqueIdentifier:`	*(type=)*

onServerGroupListEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*, *name*, *atype*, *iconID*, *saveDB*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`serverGroupID:`	*(type=)*
`name:`	*(type=)*
`atype:`	*(type=)*
`iconID:`	*(type=)*
`saveDB:`	*(type=)*

onClientChatClosedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientUniqueIdentity*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`clientID:`	*(type=)*
`clientUniqueIdentity:`	*(type=)*

onClientNeededPermissionsEvent(*self*, *serverConnectionHandlerID*, *permissionID*, *permissionValue*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`permissionID:`	*(type=)*
`permissionValue:`	*(type=)*

onServerGroupClientDeletedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientName*, *clientUniqueIdentity*, *serverGroupID*, *invokerClientID*, *invokerName*, *invokerUniqueIdentity*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`clientID:`	*(type=)*
`clientName:`	*(type=)*
`clientUniqueIdentity:`	*(type=)*
`serverGroupID:`	*(type=)*
`invokerClientID:`	*(type=)*
`invokerName:`	*(type=)*
`invokerUniqueIdentity:`	*(type=)*

onClientSelfVariableUpdateEvent(*self*, *serverConnectionHandlerID*, *flag*, *oldValue*, *newValue*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`flag:` *(type=)*

`oldValue:` *(type=)*

`newValue:` *(type=)*

onClientMoveSubscriptionEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

onPermissionListGroupEndIDEvent(*self*, *serverConnectionHandlerID*, *groupEndID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`groupEndID:` *(type=)*

onChannelGroupPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelGroupID:` *(type=)*

onChannelSubscribeFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onServerTemporaryPasswordListEvent(*self*, *serverConnectionHandlerID*, *clientNickname*, *uniqueClientIdentifier*, *description*, *password*, *timestampStart*, *timestampEnd*, *targetChannelID*, *targetChannelPW*)

Parameters

`serverConnectionHandlerID:` *(type=)*
`clientNickname:` *(type=)*
`uniqueClientIdentifier:` *(type=)*
`description:` *(type=)*
`password:` *(type=)*
`timestampStart:` *(type=)*
`timestampEnd:` *(type=)*
`targetChannelID:` *(type=)*
`targetChannelPW:` *(type=)*

onClientNeededPermissionsFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onServerGroupClientAddedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientName*, *clientUniqueIdentity*, *serverGroupID*, *invokerClientID*, *invokerName*, *invokerUniqueIdentity*)

Parameters

`serverConnectionHandlerID:` *(type=)*
`clientID:` *(type=)*
`clientName:` *(type=)*
`clientUniqueIdentity:` *(type=)*
`serverGroupID:` *(type=)*
`invokerClientID:` *(type=)*
`invokerName:` *(type=)*
`invokerUniqueIdentity:` *(type=)*

onClientIDsEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientID*, *clientName*)

Parameters

`serverConnectionHandlerID:` *(type=)*
`uniqueClientIdentifier:` *(type=)*
`clientID:` *(type=)*
`clientName:` *(type=)*

onClientMoveMovedEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *moverID*, *moverName*, *moverUniqueIdentifier*, *moveMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`moverID:` *(type=)*

`moverName:` *(type=)*

`moverUniqueIdentifier:` *(type=)*

`moveMessage:` *(type=)*

onChannelGroupPermListEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelGroupID:` *(type=)*

`permissionID:` *(type=)*

`permissionValue:` *(type=)*

`permissionNegated:` *(type=)*

`permissionSkip:` *(type=)*

onUpdateChannelEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

onClientBanFromServerEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *kickerID*, *kickerName*, *kickerUniqueIdentifier*, *time*, *kickMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`kickerID:` *(type=)*

`kickerName:` *(type=)*

`kickerUniqueIdentifier:` *(type=)*

`time:` *(type=)*

`kickMessage:` *(type=)*

onUpdateClientEvent(*self*, *serverConnectionHandlerID*, *clientID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`invokerID:` *(type=)*

`invokerName:` *(type=)*

`invokerUniqueIdentifier:` *(type=)*

onConnectionInfoEvent(*self*, *serverConnectionHandlerID*, *clientID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

onChannelPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

onAvatarUpdated(*self*, *serverConnectionHandlerID*, *clientID*, *avatarPath*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`avatarPath:` *(type=)*

onClientKickFromChannelEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *kickerID*, *kickerName*, *kickerUniqueIdentifier*, *kickMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`kickerID:` *(type=)*

`kickerName:` *(type=)*

`kickerUniqueIdentifier:` *(type=)*

`kickMessage:` *(type=)*

onHotkeyRecordedEvent(*self*, *keyword*, *key*)

Parameters

`keyword:` *(type=)*

`key:` *(type=)*

onFileListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *path*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

`path:` *(type=)*

onMessageGetEvent(*self*, *serverConnectionHandlerID*, *messageID*, *fromClientUniqueIdentity*, *subject*, *message*, *timestamp*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`messageID:` *(type=)*

`fromClientUniqueIdentity:` *(type=)*

`subject:` *(type=)*

`message:` *(type=)*

`timestamp:` *(type=)*

onChannelClientPermListEvent(*self*, *serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`clientDatabaseID:`	*(type=)*
`permissionID:`	*(type=)*
`permissionValue:`	*(type=)*
`permissionNegated:`	*(type=)*
`permissionSkip:`	*(type=)*

onConnectStatusChangeEvent(*self*, *serverConnectionHandlerID*, *newStatus*, *errorNumber*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`newStatus:`	*(type=)*
`errorNumber:`	*(type=)*

onNewChannelEvent(*self*, *serverConnectionHandlerID*, *channelID*, *channelParentID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`channelParentID:`	*(type=)*

onUpdateChannelEditedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`invokerID:`	*(type=)*
`invokerName:`	*(type=)*
`invokerUniqueIdentifier:`	*(type=)*

onChannelUnsubscribeFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*

onClientKickFromServerEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *kickerID*, *kickerName*, *kickerUniqueIdentifier*, *kickMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`kickerID:` *(type=)*

`kickerName:` *(type=)*

`kickerUniqueIdentifier:` *(type=)*

`kickMessage:` *(type=)*

onChannelGroupListFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onChannelClientPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *clientDatabaseID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

`clientDatabaseID:` *(type=)*

onClientDisplayNameChanged(*self*, *serverConnectionHandlerID*, *clientID*, *displayName*, *uniqueClientIdentifier*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`displayName:` *(type=)*

`uniqueClientIdentifier:` *(type=)*

onClientServerQueryLoginPasswordEvent(*self*, *serverConnectionHandlerID*, *loginPassword*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`loginPassword:` *(type=)*

onClientNamefromDBIDEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientDatabaseID*, *clientNickName*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`uniqueClientIdentifier:` *(type=)*

`clientDatabaseID:` *(type=)*

`clientNickName:` *(type=)*

onClientChatComposingEvent(*self*, *serverConnectionHandlerID*, *clientID*, *clientUniqueIdentity*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`clientUniqueIdentity:` *(type=)*

onPluginCommandEvent(*self*, *serverConnectionHandlerID*, *pluginName*, *pluginCommand*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`pluginName:` *(type=)*

`pluginCommand:` *(type=)*

onChannelPermListEvent(*self*, *serverConnectionHandlerID*, *channelID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

`permissionID:` *(type=)*

`permissionValue:` *(type=)*

`permissionNegated:` *(type=)*

`permissionSkip:` *(type=)*

onClientMoveTimeoutEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *timeoutMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`timeoutMessage:` *(type=)*

onChannelMoveEvent(*self*, *serverConnectionHandlerID*, *channelID*, *newChannelParentID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

`newChannelParentID:` *(type=)*

`invokerID:` *(type=)*

`invokerName:` *(type=)*

`invokerUniqueIdentifier:` *(type=)*

onClientPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *clientDatabaseID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientDatabaseID:` *(type=)*

onServerGroupPermListEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`serverGroupID:` *(type=)*

`permissionID:` *(type=)*

`permissionValue:` *(type=)*

`permissionNegated:` *(type=)*

`permissionSkip:` *(type=)*

onServerGroupListFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onPermissionOverviewEvent(*self*, *serverConnectionHandlerID*, *clientDatabaseID*, *channelID*, *overviewType*, *overviewID1*, *overviewID2*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientDatabaseID:` *(type=)*

`channelID:` *(type=)*

`overviewType:` *(type=)*

`overviewID1:` *(type=)*

`overviewID2:` *(type=)*

`permissionID:` *(type=)*

`permissionValue:` *(type=)*

`permissionNegated:` *(type=)*

`permissionSkip:` *(type=)*

onChannelPasswordChangedEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

onClientMoveEvent(*self*, *serverConnectionHandlerID*, *clientID*, *oldChannelID*, *newChannelID*, *visibility*, *moveMessage*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientID:` *(type=)*

`oldChannelID:` *(type=)*

`newChannelID:` *(type=)*

`visibility:` *(type=)*

`moveMessage:` *(type=)*

onPermissionListEvent(*self*, *serverConnectionHandlerID*, *permissionID*, *permissionName*, *permissionDescription*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`permissionID:` *(type=)*

`permissionName:` *(type=)*

`permissionDescription:` *(type=)*

onPermissionListFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onClientIDsFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

onClientNamefromUIDEvent(*self*, *serverConnectionHandlerID*, *uniqueClientIdentifier*, *clientDatabaseID*, *clientNickName*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`uniqueClientIdentifier:` *(type=)*

`clientDatabaseID:` *(type=)*

`clientNickName:` *(type=)*

onServerEditedEvent(*self*, *serverConnectionHandlerID*, *editerID*, *editerName*, *editerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`editerID:` *(type=)*

`editerName:` *(type=)*

`editerUniqueIdentifier:` *(type=)*

onChannelUnsubscribeEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

onTalkStatusChangeEvent(*self*, *serverConnectionHandlerID*, *status*, *isReceivedWhisper*, *clientID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`status:` *(type=)*

`isReceivedWhisper:` *(type=)*

`clientID:` *(type=)*

onHotkeyEvent(*self*, *keyword*)

Parameters

`keyword:` *(type=)*

onFileInfoEvent(*self*, *serverConnectionHandlerID*, *channelID*, *name*, *size*, *datetime*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`name:`	*(type=)*
`size:`	*(type=)*
`datetime:`	*(type=)*

onNewChannelCreatedEvent(*self*, *serverConnectionHandlerID*, *channelID*, *channelParentID*, *invokerID*, *invokerName*, *invokerUniqueIdentifier*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*
`channelParentID:`	*(type=)*
`invokerID:`	*(type=)*
`invokerName:`	*(type=)*
`invokerUniqueIdentifier:`	*(type=)*

onClientChannelGroupChangedEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *channelID*, *clientID*, *invokerClientID*, *invokerName*, *invokerUniqueIdentity*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelGroupID:`	*(type=)*
`channelID:`	*(type=)*
`clientID:`	*(type=)*
`invokerClientID:`	*(type=)*
`invokerName:`	*(type=)*
`invokerUniqueIdentity:`	*(type=)*

onMenuItemEvent(*self*, *serverConnectionHandlerID*, *atype*, *menuItemID*, *selectedItemID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`atype:`	*(type=)*
`menuItemID:`	*(type=)*
`selectedItemID:`	*(type=)*

onServerGroupPermListFinishedEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`serverGroupID:` *(type=)*

onClientPermListEvent(*self*, *serverConnectionHandlerID*, *clientDatabaseID*, *permissionID*, *permissionValue*, *permissionNegated*, *permissionSkip*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`clientDatabaseID:` *(type=)*

`permissionID:` *(type=)*

`permissionValue:` *(type=)*

`permissionNegated:` *(type=)*

`permissionSkip:` *(type=)*

onMessageListEvent(*self*, *serverConnectionHandlerID*, *messageID*, *fromClientUniqueIdentity*, *subject*, *timestamp*, *flagRead*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`messageID:` *(type=)*

`fromClientUniqueIdentity:` *(type=)*

`subject:` *(type=)*

`timestamp:` *(type=)*

`flagRead:` *(type=)*

onSoundDeviceListChangedEvent(*self*, *modeID*, *playOrCap*)

Parameters

`modeID:` *(type=)*

`playOrCap:` *(type=)*

onServerLogFinishedEvent(*self*, *serverConnectionHandlerID*, *lastPos*, *fileSize*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`lastPos:` *(type=)*

`fileSize:` *(type=)*

onServerGroupByClientIDEvent(*self*, *serverConnectionHandlerID*, *name*, *serverGroupList*, *clientDatabaseID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`name:` *(type=)*

`serverGroupList:` *(type=)*

`clientDatabaseID:` *(type=)*

onFileTransferStatusEvent(*self*, *transferID*, *status*, *statusMessage*, *remotefileSize*, *serverConnectionHandlerID*)

Parameters

`transferID:` *(type=)*

`status:` *(type=)*

`statusMessage:` *(type=)*

`remotefileSize:` *(type=)*

`serverConnectionHandlerID:` *(type=)*

onFileListEvent(*self*, *serverConnectionHandlerID*, *channelID*, *path*, *name*, *size*, *datetime*, *atype*, *incompletesize*, *returnCode*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`channelID:` *(type=)*

`path:` *(type=)*

`name:` *(type=)*

`size:` *(type=)*

`datetime:` *(type=)*

`atype:` *(type=)*

`incompletesize:` *(type=)*

`returnCode:` *(type=)*

onIncomingClientQueryEvent(*self*, *serverConnectionHandlerID*, *commandText*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`commandText:` *(type=)*

onServerLogEvent(*self*, *serverConnectionHandlerID*, *logMsg*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`logMsg:` *(type=)*

onComplainListEvent(*self*, *serverConnectionHandlerID*, *targetClientDatabaseID*, *targetClientNickName*, *fromClientDatabaseID*, *fromClientNickName*, *complainReason*, *timestamp*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`targetClientDatabaseID:`	*(type=)*
`targetClientNickName:`	*(type=)*
`fromClientDatabaseID:`	*(type=)*
`fromClientNickName:`	*(type=)*
`complainReason:`	*(type=)*
`timestamp:`	*(type=)*

onServerUpdatedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*

onChannelGroupListEvent(*self*, *serverConnectionHandlerID*, *channelGroupID*, *name*, *atype*, *iconID*, *saveDB*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelGroupID:`	*(type=)*
`name:`	*(type=)*
`atype:`	*(type=)*
`iconID:`	*(type=)*
`saveDB:`	*(type=)*

onChannelSubscribeEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*

onChannelDescriptionUpdateEvent(*self*, *serverConnectionHandlerID*, *channelID*)

Parameters

`serverConnectionHandlerID:`	*(type=)*
`channelID:`	*(type=)*

onServerGroupClientListEvent(*self*, *serverConnectionHandlerID*, *serverGroupID*, *clientDatabaseID*, *clientNameIdentifier*, *clientUniqueID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

`serverGroupID:` *(type=)*

`clientDatabaseID:` *(type=)*

`clientNameIdentifier:` *(type=)*

`clientUniqueID:` *(type=)*

onPermissionOverviewFinishedEvent(*self*, *serverConnectionHandlerID*)

Parameters

`serverConnectionHandlerID:` *(type=)*

Inherited from object

__delattr__(), __format__(), __getattribute__(), __hash__(), __new__(), __reduce__(), __reduce_ex__(), __repr__(), __setattr__(), __sizeof__(), __str__(), __subclasshook__()

1.1.2 Properties

Name	Description
Inherited from object	
__class__	

1.1.3 Class Variables

Name	Description
__metaclass__	**Value:** `PluginMount`
requestAutoload	**Value:** `False`
name	**Value:** `"__ts3plugin__"`
version	**Value:** `"1.0"`
apiVersion	**Value:** `20`
author	**Value:** `"Thomas \"PLuS\" Pathmann"`
description	**Value:** `"This is the baseclass for all ts3 python plugins"`
offersConfigure	**Value:** `False`
commandKeyword	**Value:** `"py"`
infoTitle	**Value:** `"pyTSon"`
menuItems	**Value:** `[(ts3defines.PLUGIN_MENU_TYPE_CLIENT, 0, "text", "icon.pn...`
hotkeys	**Value:** `[("keyword", "description")]`

2 Module pytsonui

2.1 Functions

getValues(*parent*, *title*, *params*, *doneclb*)

Convenience function to open a dialog to get multiple input values from the user.

Parameters

`parent:` the dialog's parent, pass None to ignore

(type=QWidget (or derived type))

`title:` the dialog's title

(type=str)

`params:` a dict definining the user input type {'key': (ValueType, label, startingValue, minimum, maximum)}. Potential types are defined in the enum ValueType. The label will be displayed right next to the input widget. All other elements in this tuple are dependent on the ValueType. startingValue defines a predefined value of input. Minimum and maximum define input limits. boolean: startingValue is bool, minimum and maximum are not used; the widget used is a QCheckBox without an extra QLabel integer: startingValue, minimum and maximum are int; the widget used is a QSpinBox with an extra QLabel double: startingValue, minimum; the widget used is a QDoubleSpinBox with an extra QLabel string: startingValue is str, minimum is not used, if maximum == 1 the widget used is a QLineEdit, otherwise a QPlainTextEdit with a maximumBlockCount of maximum, each with an extra QLabel listitem: startingValue is a tuple([str], [int]) defining the listitems in the first element, the second element is a list with prechecked item indexes, minimum is an int defining how much items the user at least has to choose, maximum is an int defining if the user can choose more than one item (maximum != 1), depending on minimum and maximum the used widget is a QGroupBox and multiple QRadioButtons, a QComboBox with an extra QLabel or a QListWidget with an extra QLabel

(type=dict{str: tuple(ValueType, str, int/double/str/tuple(list[str], list[int]), int/double, int/double)})

`doneclb:` a callable which gets the dialogs return code (see QDialog.DialogCode) and on success, a dict with the resulting values, referenced by the key.

(type=callable(int, dict{str: int/str/bool/[str]}))

Return Value

connectSignalSlotsByName(*sender*, *receiver*)

Connects pythonqt signals by name
(receiver.on_<sender.objectname>_<signalname>)

Parameters

`sender:` the sender of signals

(type=QObject)

`receiver:` the receiver which has slots as callables defined

(type=object)

retrieveWidgets(*obj*, *parent*, *widgets*)

Retrieves widgets from a list and adds them as attribute to another object. If defined, signals from widgets are connected by name to methods in obj.

Parameters

`obj:` the object which will get the attributes added

(type=object)

`parent:` the toplevel widget

(type=QWidget)

`widgets:` a recursive (parent-relation of widgets) list of tuples, defining which widgets should be added as attributes to obj. The elements must be children of parent. First element of tuple must held the widget's objectname. If second element is True, the widget will be added as property (by objectname) to obj. Third element of the tuple are the child widgets, which should be handled by setupui

(type=list[tuple(str, bool, list(...))])

retrieveAllWidgets(*obj*, *parent*)

Retrieves all child widgets from a parent widget and adds them as attribute to another object. If defined, signals from widgets are connected by name to methods in obj.

Parameters

`obj:` the object which will get the attributes added

(type=object)

`parent:` the toplevel widget

(type=QWidget)

setupUi(*obj*, *uipath*, *widgets*=`None`)

Loads a Qt designer file (.ui), creates the widgets defined in and adds them as property to a given object. This internally calls retrieveWidgets, so signals from widgets are connected by name to obj.

Parameters

`obj:` The object which will act as parent of the loaded ui (this object will receive a new layout)

(type=QWidget)

`uipath:` the path to the Qt designer file

(type=str)

`widgets:` optional argument; a recursive (parent-relation of widgets) list of tuples, defining which widgets should be added as attributes to obj. See retrieveWidgets for details. If you omit this or pass None, recursively all child widgets will be stored

(type=list[tuple(str, bool, list(...))] or None)

defaultFont()

2.2 Class ValueType

EnumMeta('Enum', (object,), temp_enum_dict) ┐

pytsonui.ValueType

enum to define the types of value shown and received with getValues.

2.2.1 Class Variables

Name	Description
boolean	**Value:** `1`
integer	**Value:** `2`
double	**Value:** `3`
string	**Value:** `4`
listitem	**Value:** `5`

2.3 Class ConfigurationDialog

??-7

pytsonui.ConfigurationDialog

2.3.1 Methods

__init__(*self*, *cfg*, *host*, *parent*=`None`)

setupList(*self*)

setupValues(*self*)

setupSlots(*self*)

onDifferentApiButtonChanged(*self*, *state*)

onPluginsListCurrentItemChanged(*self*, *cur*, *prev*)

onPluginsListItemChanged(*self*, *item*)

onReloadButtonClicked(*self*)

onSettingsButtonClicked(*self*)

onBackgroundColorButtonClicked(*self*)

onTextColorButtonClicked(*self*)

onFontFamilyComboChanged(*self*, *font*)

onFontSizeSpinChanged(*self*, *size*)

onTabcompleteButtonChanged(*self*, *state*)

onSpacesButtonChanged(*self*, *state*)

onTabwidthSpinChanged(*self*, *width*)

on_scriptButton_clicked(*self*)

on_scriptEdit_textEdited(*self*, *text*)

on_silentButton_toggled(*self*, *act*)

2.3.2 Class Variables

Name	Description
CONF_WIDGETS	**Value:** `[("tabWidget", False, [("pluginsTab", False, [("different...`

2.4 Class StdRedirector

2.4.1 Methods

__init__(*self*, *callback*)

write(*self*, *text*)

2.5 Class PythonConsole

??-6

pytsonui.PythonConsole

2.5.1 Methods

__init__(*self*, *tabcomplete*=`True`, *spaces*=`True`, *tabwidth*=`2`, *font*=`defaultFont()`, *bgcolor*=`Qt.black`, *textcolor*=`Qt.white`, *width*=`800`, *height*=`600`, *startup*=`""`, *silentStartup*=`False`, *parent*=`None`)

setFont(*self*, *f*)

prompt(*self*)

promptLength(*self*)

writePrompt(*self*, *newline*)

promptCursor(*self*)

keyPressEvent(*self*, *e*)

mousePressEvent(*self*, *e*)

mouseReleaseEvent(*self*, *e*)

doKeyboardInterrupt(*self*)

doEndFile(*self*)

currentLine(*self*)

removeCurrentLine(*self*)

addHistory(*self*, *cmd*)

doHistoryUp(*self*)

doHistoryDown(*self*)

doTab(*self*)

doUntab(*self*)

appendLine(*self*, *text*)

runCommand(*self*, *cmd*, *silent*)

doExecuteCommand(*self*)

3 Module ts3module

3.1 Class ts3

3.1.1 Methods

getPluginID()

Returns pyTSon's plugin id

Return Value

the plugin id

(type=string)

getProfileList(*profile*)

Returns a list of existing profiles and the default profile's index in list.

Parameters

`profile:` the profile type, see ts3defines.PluginGuiProfile

(type=int)

Return Value

a tuple, containing the errorcode, the default profile's index and the profile list

(type=tuple (int, int, [string]))

getPreProcessorInfoValueFloat(*serverConnectionHandlerID*, *ident*)

Queries a sound preprocessor flag and returns it as float.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`ident:` the flag to be queried

(type=string)

Return Value

A tuple, containing the errorcode and the value of the queried flag

(type=tuple (int, float))

requestFileList(*serverConnectionHandlerID*, *channelID*, *channelPW*, *path*, *returnCode*)

Requests the filelist of a channel. The events onFileListEvent and onFileListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`channelPW:`	the password of the channel, pass an empty string if the channel is not password protected *(type=string)*
`path:`	the path of the directory to be listed, pass '/' for the root path *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

getChannelOfClient(*serverConnectionHandlerID*, *clientID*)

Returns the channel of a client.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client *(type=int)*

Return Value

a tuple, containing the errorcode and the channel

(type=tuple (int, int))

requestRenameFile(*serverConnectionHandlerID*, *fromChannelID*, *channelPW*, *toChannelID*, *toChannelPW*, *oldFile*, *newFile*, *returnCode*)

Renames a file or moves it to another channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`fromChannelID:` the ID of the channel, the file is currently placed in

(type=int)

`channelPW:` the password of the channel, the file is currently placed in, pass an empty string if channel is not password protected

(type=string)

`toChannelID:` //FIXME: pass 0, if not moving, just renaming?

(type=int)

`toChannelPW:` the password of the channel, to which the file should move to, pass an empty string if channel is not password protected or //FIXME: pass empty string if not moving

(type=string)

`oldFile:` the complete path to the file

(type=string)

`newFile:` the complete path to the new filename

(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

getTransferFileName(*transferID*)

Returns the filename of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer

(type=int)

Return Value

a tuple, containing the errorcode and the filename

(type=tuple (int, string))

acquireCustomPlaybackData(*deviceName*, *samples*)

Retrieves playback data from the clientlib

Parameters

`deviceName:` the name of the playback device previously registered with registerCustomDevice

(type=string)

`samples:` specifies how long the resultbuffer should be, which is passed to the clientlib

(type=int)

Return Value

the errorcode

(type=int)

requestChannelGroupDel(*serverConnectionHandlerID*, *channelGroupID*, *force*, *returnCode*)

Deletes a channelgroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelGroupID:` the ID of the channelgroup
(type=int)

`force:` if set to 1 (or True), even if there are users assigned to this channelgroup, it will be deleted //FIXME: right?
(type=int or bool)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

setPreProcessorConfigValue(*serverConnectionHandlerID*, *ident*, *value*)

Sets a sound preprocessor flag.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`ident:` the flag to be set
(type=string)

`value:` the value to set the flag to
(type=string)

Return Value

the errorcode
(type=int)

getHotkeyFromKeyword(*keywords*)

Parameters

`keywords:` *(type=)*

Return Value

(type=)

getServerVersion(*serverConnectionHandlerID*)

Returns the server version.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

Return Value

the server version

(type=int)

getConfigPath(*maxLen*=`256`)

Returns the ts3 config path.

Parameters

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

the config path

(type=string)

getCurrentPlayBackMode(*serverConnectionHandlerID*)

Queries the current playback mode on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection

(type=int)

Return Value

A tuple, containing the errorcode and the current playback mode

(type=tuple (int, string))

requestMessageAdd(*serverConnectionHandlerID*, *toClientUID*, *subject*, *message*, *returnCode*)

Sends an offline message to another user.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`toClientUID:`	the UID of the user *(type=string)*
`subject:`	the subject of the message *(type=string)*
`message:`	the message *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

banclientdbid(*serverConnectionHandlerID*, *clientDBID*, *timeInSeconds*, *banReason*, *returnCode*)

Bans a user defined by his database ID.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientDBID:` the database ID of the user
(type=int)

`timeInSeconds:` the time, the client should be banned for, pass 0 to add a permanent ban
(type=int)

`banReason:` the reason for the ban
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getClientID(*serverConnectionHandlerID*)

Returns the own client ID on a given serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

a tuple, containing the errorcode and the client ID
(type=tuple (int, int))

printMessageToCurrentTab(*message*)

Prints a message to the currently visible tab.

Parameters

`message:` the message to send

(type=string)

getBookmarkList()

Returns the list of bookmarks.

Return Value

a tuple, containing the errorcode and a list of tuples (name, isFolder, uid, childs)

(type=tuple (int, [tuple (string, int or bool, string or None, [childs])]))

closePlaybackDevice(*serverConnectionHandlerID*)

Closes a playback device on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

Return Value

the errorcode

(type=int)

playWaveFileHandle(*serverConnectionHandlerID*, *path*, *loop*)

Plays a wavefile sound on a serverconnection and returns a handle to it.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection on which the sound will be played on

(type=int)

`path:` the path to the wavefile on the system

(type=string)

`loop:` if set to 1 (or True), the sound will loop

(type=int or bool)

Return Value

A tuple, containing the errorcode and the handle, with which the sound can be paused and unpaused

(type=tuple (int, int))

getDefaultCaptureDevice(*modeID*)

Queries the default capture device.

Parameters

`modeID:` Defines the capture mode to use

(type=string)

Return Value

A tuple, containing the errorcode and the default capture device as tuple (devicename, deviceid)

(type=tuple (int, (string, string)))

requestServerGroupDelPerm(*serverConnectionHandlerID*, *serverGroupID*, *continueOnError*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`continueOnError:` if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled
(type=int or bool)

`permissionIDArray:` list of permission IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

flushClientSelfUpdates(*serverConnectionHandlerID*, *returnCode*)

Flushes the changes made by the setClientSelfVariable-functions to the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getTransferStatus(*transferID*)

Returns the status of a filetransfer, whether if it is initialising, active or finished see ts3defines.FileTransferState

Parameters

`transferID:` the ID of the filetransfer
(type=int)

Return Value

a tuple, containing the errorcode and the status
(type=tuple (int, int))

getClientVariableAsUInt64(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as unsigned long long int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

requestPermissionList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of permissions available on the server. The events onPermissionListEvent and onPermissionListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestUnmuteClients(*serverConnectionHandlerID*, *clientIDArray*, *returnCode*)

Unmutes a list of clients.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientIDArray:`	a list of client IDs *(type=list [int])*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestClientSetWhisperList(*serverConnectionHandlerID*, *clientID*, *targetChannelIDArray*, *targetClientIDArray*, *returnCode*)

Modifies the whisper list of a client.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client whose whisperlist is modified. If set to 0, the own whisper list is modified *(type=int)*
`targetChannelIDArray:`	a list of channel IDs the client will whisper to *(type=list [int])*
`targetClientIDArray:`	a list of client IDs the client will whisper to *(type=list [int])*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

openCaptureDevice(*serverConnectionHandlerID*, *modeID*, *captureDevice*)

Opens a playback device on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection on which the capture device should be initialized on

(type=int)

`modeID:` the playback mode to use

(type=string)

`captureDevice:` the id of the capture device

(type=string)

Return Value

the errorcode

(type=int)

cleanUpConnectionInfo(*serverConnectionHandlerID*, *clientID*)

//FIXME:

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`clientID:` the ID of the client

(type=int)

Return Value

the errorcode

(type=int)

getServerVariableAsInt(*serverConnectionHandlerID*, *flag*)

Returns a server variable as int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

closeWaveFileHandle(*serverConnectionHandlerID*, *waveHandle*)

Closes a wavefile sound handle previously returned by playWaveFileHandle.

Parameters

`serverConnectionHandlerID:` the ID of the serverConnection the sound was played on
(type=int)

`waveHandle:` the handle returned by playWaveFileHandle
(type=int)

Return Value

the errorcode
(type=int)

clientChatComposing(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Sends the client chat composing command to a client the own client is currently chatting with.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client, the own client is chatting with *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

createBookmark(*bookmarkuuid*, *serverLabel*, *serverAddress*, *serverPassword*, *nickname*, *channel*, *channelPassword*, *captureProfile*, *playbackProfile*, *hotkeyProfile*, *soundProfile*, *uniqueUserId*, *oneTimeKey*, *phoneticName*)

Creates a new bookmark.

Parameters

`bookmarkuuid:` //FIXME: parent?
(type=string)

`serverLabel:` the label of the connection
(type=string)

`serverAddress:` host or ip address
(type=string)

`serverPassword:` password to the server, pass an empty string if the server is not password protected
(type=string)

`nickname:` the user's nickname
(type=string)

`channel:` complete path to the channel to connect to
(type=string)

`channelPassword:` password to the channel, pass an empty string if the channel is not password protected
(type=string)

`captureProfile:` the name of the capture profile to use
(type=string)

`playbackProfile:` the name of the playback profile to use
(type=string)

`hotkeyProfile:` the name of the hotkey profile to use
(type=string)

`soundProfile:` the name of the sound profile to use
(type=string)

`uniqueUserId:` identity to use
(type=string)

`oneTimeKey:` privilege key to use on connect
(type=string)

`phoneticName:` phonetic nickname
(type=string)

Return Value

the errorcode
(type=int)

requestClientSetIsTalker(*serverConnectionHandlerID*, *clientID*, *isTalker*, *returnCode*)

Grants or revokes the talker flag of a client.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client *(type=int)*
`isTalker:`	if set to 1 (or True) grants talker flag, if 0 (or False) revokes talker flag *(type=int or bool)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

systemset3DListenerAttributes(*serverConnectionHandlerID*, *position*, *forward*, *up*)

Sets the position, velocity and orientation of the own client in 3D space

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`position:` A tuple defining the 3D position, pass None to ignore
(type=tuple (float, float, float))

`forward:` A tuple defining the forward orientation of the listener. The vector must be of unit length and perpendicular to the up vector. Pass None to ignore.
(type=tuple (float, float, float))

`up:` A tuple defining the upward orientation of the listener. The vector must be of unit length and perpendicular to the forward vector. Pass None to ignore.
(type=tuple (float, float, float))

Return Value

the errorcode
(type=int)

getChannelList(*serverConnectionHandlerID*)

Returns all channels on the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

a tuple, containing the errorcode and a list of channel IDs
(type=tuple (int, [int]))

requestServerVariables(*serverConnectionHandlerID*)

Requests all server variables of a serverconnection. The event onServerUpdatedEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

the errorcode
(type=int)

getCaptureModeList()

Queries all available capture modes.

Return Value

A tuple, containing the errorcode and the list of capture modes
(type=tuple (int, [string]))

requestServerTemporaryPasswordList(*serverConnectionHandlerID*, *returnCode*)

Requests a list of existing temporary passwords. The event onServerTemporaryPasswordListEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestMessageDel(*serverConnectionHandlerID*, *messageID*, *returnCode*)

Deletes an offline message.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`messageID:` the ID of the message
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getEncodeConfigValue(*serverConnectionHandlerID*, *ident*)

Queries a speex encoder option.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`ident:` the encoder option to be queried
(type=string)

Return Value

A tuple, containing the errorcode and the flag's value
(type=tuple (int, string))

getClientDisplayName(*serverConnectionHandlerID*, *clientID*, *maxLen*=128)

Returns the client display name receiving from the client's contacts settings.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 128
(type=int)

Return Value

a tuple, containing the errorcode and the display name
(type=tuple (int, string))

unregisterCustomDevice(*deviceID*)

Unregisters a custom device, previously registered with registerCustomDevice.

Parameters

`deviceID:` the ID of the device, used in registerCustomDevice
(type=string)

Return Value

the errorcode
(type=int)

urlsToBB(*text*, *maxLen*=256)

Converts an url to the BB-code respresentation.

Parameters

`text:` the url

(type=string)

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

the BB-code representation

(type=string)

requestChannelDescription(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Requests the channel description of a channel. Afterwards, getChannelVariableAsString can return it.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelID:` the ID of the channel

(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

stopConnection(*serverConnectionHandlerID*, *quitMessage*)

Stops the connection of a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`quitMessage:` a message displayed when leaving the server encoded in UTF-8
(type=string)

Return Value

the errorcode
(type=int)

requestClientKickFromServer(*serverConnectionHandlerID*, *clientID*, *kickReason*, *returnCode*)

Kicks a client from the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client to kick
(type=int)

`kickReason:` the reason for the kick
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

(type=)

requestServerGroupDelClient(*serverConnectionHandlerID*, *serverGroupID*, *clientDatabaseID*, *returnCode*)

Deletes a user defined by his database ID from a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getTransferFilePath(*transferID*)

Returns the filepath of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer
(type=int)

Return Value

a tuple, containing the errorcode and the filepath
(type=tuple (int, string))

requestServerGroupDel(*serverConnectionHandlerID*, *serverGroupID*, *force*, *returnCode*)

Deletes a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`force:` if set to 1 (or True), even if there are users assigned to this servergroup, it will be deleted //FIXME: right?
(type=int or bool)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

(type=)

getChannelClientList(*serverConnectionHandlerID*, *channelID*)

Returns all clients in a specified channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

Return Value

a tuple, containing the errorcode and a list of client IDs or None if the call failed
(type=tuple (int, [int]) or tuple(int, None))

getConnectionVariableAsDouble(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns a client's connection variable as python floating point variable.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, double))

requestMuteClients(*serverConnectionHandlerID*, *clientIDArray*, *returnCode*)

Mutes a list of clients.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientIDArray:` a list of client IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getCurrentTransferSpeed(*transferID*)

Returns the current transfer speed of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer

(type=int)

Return Value

a tuple, containing the errorcode and the speed

(type=tuple (int, float))

requestSendClientQueryCommand(*serverConnectionHandlerID*, *command*, *returnCode*)

//FIXME:

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`command:` the command to send

(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

getPlaybackConfigValueAsFloat(*serverConnectionHandlerID*, *ident*)

Queries a playback option.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`ident:` the playback option to be queried
(type=string)

Return Value

A tuple, containing the errorcode and the flag's value
(type=tuple (int, float))

getClientSelfVariableAsString(*serverConnectionHandlerID*, *flag*)

Returns the value of a given flag of the own client as string.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the queried flag
(type=tuple (int, string))

getDefaultPlaybackDevice(*modeID*)

Queries the default playback device.

Parameters

`modeID:` Defines the playback mode to use
(type=string)

Return Value

A tuple, containing the errorcode and the default playback device as tuple (devicename, deviceid)
(type=tuple (int, (string, string)))

closeCaptureDevice(*serverConnectionHandlerID*)

Closes a capture device on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

Return Value

the errorcode

(type=int)

isTransferSender(*transferID*)

//FIXME:

Parameters

`transferID:` the ID of the filetransfer

(type=int)

Return Value

a tuple, containing the errorcode and //FIXME:

(type=tuple (int, int or bool))

clientChatClosed(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *clientID*, *returnCode*)

Sends the client chat closed command to a client the own client is currently chatting with.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`clientUniqueIdentifier:` the uid of the own chatting client

(type=string)

`clientID:` the ID of the client, the own client is chatting with

(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

guiConnectBookmark(*connectTab*, *bookmarkuuid*)

Connects to a server from a bookmark and displays it as tab in the client.

Parameters

`connectTab:` defines, which tab will be used, see ts3defines.PluginConnectTab

(type=int)

`bookmarkuuid:` UID of the bookmark

(type=string)

Return Value

a tuple, containing the errorcode and the ID of the created serverconnection handler

(type=tuple (int, int))

setPlaybackConfigValue(*serverConnectionHandlerID*, *ident*, *value*)

Sets a playback option.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`ident:` the playback option to reset
(type=string)

`value:` the value to set
(type=string)

Return Value

the errorcode
(type=int)

requestFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *overwrite*, *resume*, *destinationDirectory*, *returnCode*)

Starts a filedownload from the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelID:` the ID of the channel in which the file is placed in

(type=int)

`channelPW:` the password of the channel, pass an empty string if the channel is not password protected

(type=string)

`file:` the complete path to the file in the channel

(type=string)

`overwrite:` if set to 1 (or True) and a file with that name already exists will be overwritten

(type=int or bool)

`resume:` if set to 1 (or True), a previously started filetransfer can be resumed

(type=int or bool)

`destinationDirectory:` the path to the directory, where the downloaded fill will be placed in

(type=strin)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

a tuple, containing the errorcode and the ID of the filetransfer

(type=tuple (int, int))

serverPropertyStringToFlag(*serverPropertyString*)

//FIXME:

Parameters

`serverPropertyString:` *(type=)*

Return Value

a tuple, containing the errorcode and

(type=tuple (int, int))

requestClientDelPerm(*serverConnectionHandlerID*, *clientDatabaseID*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a user.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`clientDatabaseID:` the database ID of the user

(type=int)

`permissionIDArray:` a list of permission IDs

(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

initiateGracefulPlaybackShutdown(*serverConnectionHandlerID*)

Graceful shutdown the playback device on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

Return Value

the errorcode

(type=int)

requestChannelPermList(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Requests the list of permissions assigned to a channel. The events onChannelPermListEvent and onChannelPermListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelMove(*serverConnectionHandlerID*, *channelID*, *newChannelParentID*, *newChannelOrder*, *returnCode*)

Moves a channel to a new parent channel.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel to move *(type=int)*
`newChannelParentID:`	the ID of the new parent channel *(type=int)*
`newChannelOrder:`	Channel order defining where the channel should be sorted under the new parent. Pass 0 to sort the channel right after the parent *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestServerGroupAdd(*serverConnectionHandlerID*, *groupName*, *groupType*, *returnCode*)

Adds a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`groupName:` the name of the group to create
(type=string)

`groupType:` type of the servergroup, see ts3defines.GroupType
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestMessageList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of offline messages. The event onMessageListEvent will be triggered. //FIXME: onMessageListFinishedEvent fehlt?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getClientList(*serverConnectionHandlerID*)

Returns all clients in view on the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

a tuple, containing the errorcode and the list of client IDs
(type=tuple (int, [int]))

requestComplainAdd(*serverConnectionHandlerID*, *targetClientDatabaseID*, *complainReason*, *returnCode*)

Adds a complain to a user defined by his database ID.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`targetClientDatabaseID:` the database ID of the user
(type=int)

`complainReason:` the reason for the complain
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestCreateDirectory(*serverConnectionHandlerID*, *channelID*, *channelPW*, *directoryPath*, *returnCode*)

Creates a directory in a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`channelPW:` the password of the channel, pass an empty string if channel is not password protected
(type=string)

`directoryPath:` the complete path of the to be created directory
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

(type=)

getChannelVariableAsUInt64(*serverConnectionHandlerID*, *channelID*, *flag*)

Returns a channel variable as unsigned long long int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

requestIsTalker(*serverConnectionHandlerID*, *isTalkerRequest*, *isTalkerRequestMessage*, *returnCode*)

Requests talk power or revokes the talk power request.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`isTalkerRequest:`	if set to 1 (or True) requests talk power, if 0 (or False) revokes the talk power request *(type=int or bool)*
`isTalkerRequestMessage:`	the message of the request *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestComplainList(*serverConnectionHandlerID*, *targetClientDatabaseID*, *returnCode*)

Requests the list of complains to a user. The event onComplainListEvent will be triggered. //FIXME: ts3plugin_onComplainListFinishedEvent is missing?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`targetClientDatabaseID:` the database ID of the user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getChannelConnectInfo(*serverConnectionHandlerID*, *channelID*, *maxLen*)

Returns the channel connect info (path and password) of a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256
(type=int)

Return Value

a tuple, containing the errorcode, the path and the password of a channel
(type=tuple (int, string, string))

requestClientVariables(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Requests latest data for a given client. The event onUpdateClientEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelClientAddPerm(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionIDArray*, *permissionValueArray*, *returnCode*)

Adds a list of permissions on a channel to a user.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`permissionIDArray:` a list of permission IDs
(type=list [int])

`permissionValueArray:` list of permission values, in order of the permissions in permissionIDArray
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestChannelSubscribe(*serverConnectionHandlerID*, *channelIDArray*, *returnCode*)

Subscribes to a list of channels to get notifications of the clients in them.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelIDArray:` a list of channel IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getAppPath(*maxLen*=`256`)

Returns the ts3 application path.

Parameters

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256
(type=int)

Return Value

the application path
(type=string)

requestClientIDs(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *returnCode*)

Requests the client IDs for a given UID. Will trigger the event onClientIDsEvent.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientUniqueIdentifier:`	the UID of the client *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelSubscribeAll(*serverConnectionHandlerID*, *returnCode*)

Subscribes to all channels on the server.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

channelPropertyStringToFlag(*channelPropertyString*)

//FIXME:

Parameters

`channelPropertyString:` *(type=string)*

Return Value

a tuple, containing the errorcode and

(type=tuple (int, int))

getServerVariableAsUInt64(*serverConnectionHandlerID*, *flag*)

Returns a server variable as unsigned long long int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag

(type=tuple (int, int))

banclient(*serverConnectionHandlerID*, *clientID*, *timeInSeconds*, *banReason*, *returnCode*)

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`timeInSeconds:` the time, the client should be banned for, pass 0 to add a permanent ban
(type=int)

`banReason:` the reason for the ban
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getClientVariableAsInt(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

setClientVolumeModifier(*serverConnectionHandlerID*, *clientID*, *value*)

Sets the volume modifier of a client.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the client's ID
(type=int)

`value:` the value to set
(type=float)

Return Value

the errorcode
(type=int)

getClientVariableAsString(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns the value of a given flag of a client as string value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, string))

startConnection(*serverConnectionHandlerID*, *identity*, *ip*, *port*, *nickname*, *defaultChannelArray*, *defaultChannelPassword*, *serverPassword*)

Starts a connection to the given server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`identity:` the client's identity
(type=string)

`ip:` hostname or ip of the server
(type=string)

`port:` port of the server
(type=int)

`nickname:` the client's nickname
(type=string)

`defaultChannelArray:` list of strings defining the path to a channel on the server, pass empty list to join in server's default channel
(type=list [string])

`defaultChannelPassword:` password of the default channel, pass an empty string if not using defaultChannelArray or channel is not password protected
(type=string)

`serverPassword:` password of the server, pass an empty string if the server is not password protected
(type=string)

Return Value

the errorcode
(type=int)

getCurrentCaptureDeviceName(*serverConnectionHandlerID*)

Queries the current playback device's name on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection

(type=int)

Return Value

A tuple, containing the errorcode, the capture device's name and the status, if it's default

(type=tuple (int, string, int))

requestSetClientChannelGroup(*serverConnectionHandlerID*, *channelGroupIDArray*, *channelIDArray*, *clientDatabaseIDArray*, *returnCode*)

Adds a list of users to a list of channelgroups.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelGroupIDArray:` a list of channelgroup IDs

(type=list [int])

`channelIDArray:` a list of channel IDs

(type=list [int])

`clientDatabaseIDArray:` a list of client database IDs

(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

verifyChannelPassword(*serverConnectionHandlerID*, *channelID*, *channelPassword*, *returnCode*)

Verifies the password to a channel. //FIXME: serverErrorEvent oder errorcode?

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`channelPassword:`	the password to be verified *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

getChannelIDFromChannelNames(*serverConnectionHandlerID*, *channelNameArray*)

Returns the ID of a channel defined by its name.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelNameArray:`	list of strings, defining the position of the channel (['grandparent', 'parent', 'channel']) *(type=list [string])*

Return Value

a tuple, containing the errorcode and the ID of the channel

(type=tuple (int, int))

requestChannelGroupPermList(*serverConnectionHandlerID*, *channelGroupID*, *returnCode*)

Requests the list of permissions assigned to a channelgroup. The events onChannelGroupPermListEvent and onChannelGroupPermListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelGroupID:`	the ID of the channelGroupID *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

stopVoiceRecording(*serverConnectionHandlerID*)

Stops voice recording on a serverconnection

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*

Return Value

the errorcode

(type=int)

getAvatar(*serverConnectionHandlerID*, *clientID*, *maxLen*=256)

Returns the path on the system to the avatar image file of a client.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256
(type=int)

Return Value

a tuple, containing the errorcode and the path to the avatar
(type=tuple (int, string))

isReceivingWhisper(*serverConnectionHandlerID*, *clientID*)

Returns the status of a client whether he accepts whispering to him.
//FIXME: depending on my and his permissions?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

Return Value

a tuple, containing the errorcode and the status
(type=tuple (int, int or bool))

requestChannelDelPerm(*serverConnectionHandlerID*, *channelID*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`permissionIDArray:` a list of permission IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestServerTemporaryPasswordAdd(*serverConnectionHandlerID*, *password*, *description*, *duration*, *targetChannelID*, *targetChannelPW*, *returnCode*)

Adds a temporary password to the server.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`password:`	the temporary password *(type=string)*
`description:`	the description of the temporary password *(type=string)*
`duration:`	the duration in seconds *(type=int)*
`targetChannelID:`	the ID of the channel to which the accessing clients will join by default *(type=int)*
`targetChannelPW:`	the password of the targetChannel, pass an empty string, if the channel is not password protected *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestServerGroupsByClientID(*serverConnectionHandlerID*, *clientDatabaseID*, *returnCode*)

Requests all servergroups of a user defined by his database ID. The event onServerGroupByClientIDEvent will be triggered. //FIXME: ts3plugin_onServerGroupByClientIDFinishedEvent is missing?

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientDatabaseID:`	the database ID of the user *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

getClientSelfVariableAsInt(*serverConnectionHandlerID*, *flag*)

Returns the value of a given flag of the own client as int value.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`flag:`	the flag to return *(type=int)*

Return Value

a tuple, containing the errorcode and the value of the queried flag

(type=tuple (int, int))

getServerVariableAsString(*serverConnectionHandlerID*, *flag*)

Returns a server variable as string value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, string))

getAverageTransferSpeed(*transferID*)

Returns the average transfer speed of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer
(type=int)

Return Value

a tuple, containing the errorcode and the speed
(type=tuple (int, float))

requestMessageUpdateFlag(*serverConnectionHandlerID*, *messageID*, *flag*, *returnCode*)

//FIXME: wäh?

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`messageID:`	the ID of the message *(type=int)*
`flag:`	*(type=)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

setClientSelfVariableAsString(*serverConnectionHandlerID*, *flag*, *value*)

Sets a variable of the own client to a new string value.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`flag:`	the flag to set *(type=int)*
`value:`	the new value *(type=string)*

Return Value

the errorcode

(type=int)

getConnectionVariableAsUInt64(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns a client's connection variable as unsigned long long int variable.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

registerCustomDevice(*deviceID*, *deviceDisplayName*, *capFrequency*, *capChannels*, *playFrequency*, *playChannels*)

Registers a custom device, announcing the device ID and name to the Client Lib.

Parameters

`deviceID:` ID string of the custom device, under which the device can be later accessed

(type=string)

`deviceDisplayName:` Displayed name of the custom device. Freely choose a name which identifies your device

(type=string)

`capFrequency:` Frequency of the capture device

(type=int)

`capChannels:` Number of channels of the capture device. This value depends on if the used codec is a mono or stereo CodecEncryptionMode

(type=int)

`playFrequency:` Frequency of the playback deviceDisplayName

(type=int)

`playChannels:` Number of channels of the playback device

(type=int)

Return Value

the errorcode

(type=int)

verifyServerPassword(*serverConnectionHandlerID*, *serverPassword*, *returnCode*)

Verifies the password to a server. //FIXME: serverErrorEvent oder errorcode?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverPassword:` the password to be verified
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getParentChannelOfChannel(*serverConnectionHandlerID*, *channelID*)

Returns the parent channel of another channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

Return Value

a tuple, containing the errorcode and the ID of the parent channel
(type=tuple (int, int))

setClientSelfVariableAsInt(*serverConnectionHandlerID*, *flag*, *value*)

Sets a variable of the own client to a new int value.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`flag:`	the flag to set *(type=int)*
`value:`	the new value *(type=int)*

Return Value

the errorcode

(type=int)

requestChannelGroupList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of channelgroups. The events onChannelGroupListEvent and onChannelGroupListEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

flushChannelUpdates(*serverConnectionHandlerID*, *channelID*, *returnCode*)

Flushes the changes made by the setChannelVariable-functions to the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channelID
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getCurrentServerConnectionHandlerID()

Returns the current serverconnection handler.

Return Value

the ID of the current serverconnection handler
(type=int)

processCustomCaptureData(*deviceName*, *buffer*)

Sends captured data from a custom device to the client libg

Parameters

`deviceName:` the name of the device capturing the data, previously registered with registerCustomDevice
(type=string)

`buffer:` a list containing the buffered data
(type=list [int])

Return Value

the errorcode
(type=int)

getResourcesPath(*maxLen*=256)

Returns the ts3 resources path.

Parameters

`maxLen`: length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

the resources path

(type=string)

requestComplainDelAll(*serverConnectionHandlerID*, *targetClientDatabaseID*, *returnCode*)

Deletes all complains to a user.

Parameters

`serverConnectionHandlerID`: the ID of the serverconnection

(type=int)

`targetClientDatabaseID`: the database ID of the user

(type=int)

`returnCode`: returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

requestChannelAddPerm(*serverConnectionHandlerID*, *channelID*, *permissionIDArray*, *permissionValueArray*, *returnCode*)

Adds a list of permissions to a channel.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`permissionIDArray:`	a list of permission IDs *(type=list [int])*
`permissionValueArray:`	list of permission values, in order of the permissions in permissionIDArray *(type=list [int])*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

haltTransfer(*serverConnectionHandlerID*, *transferID*, *deleteUnfinishedFile*, *returnCode*)

Halts a currently running filetransfer.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`transferID:`	the ID of the filetransfer *(type=int)*
`deleteUnfinishedFile:`	if set to 1 (or True) and the file is not yet finished, it will be deleted; to prevent, pass 0 (or False) *(type=int or bool)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

(type=)

requestServerGroupPermList(*serverConnectionHandlerID*, *serverGroupID*, *returnCode*)

Requests the list of permissions assigned to a servergroup. The events onServerGroupPermListEvent and onServerGroupPermListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`serverGroupID:`	the ID of the servergroup *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

logMessage(*logMessage*, *severity*, *channel*, *logID*)

Logs a string.

Parameters

`logMessage:`	Text which should be logged *(type=string)*
`severity:`	The level of the message, warning or error. Defined by the class LogLevel *(type=int)*
`channel:`	Custom text to categorize the message channel *(type=string)*
`logID:`	ID of the serverconnection to identify the current server connection when using multiple connections, 0 if unused *(type=int)*

Return Value

The errorcode

(type=int)

getChannelVariableAsString(*serverConnectionHandlerID*, *channelID*, *flag*)

Returns a channel variable as string value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

requestChannelGroupAdd(*serverConnectionHandlerID*, *groupName*, *groupType*, *returnCode*)

Adds a channelgroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`groupName:` the name of the channelgroup to create
(type=string)

`groupType:` type of the channelgroup, see ts3defines.GroupType
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getChannelVariableAsInt(*serverConnectionHandlerID*, *channelID*, *flag*)

Returns a channel variable as int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`flag:` the flag to return
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, int))

requestClientKickFromChannel(*serverConnectionHandlerID*, *clientID*, *kickReason*, *returnCode*)

Kicks a client from its current channel to the default one.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client to kick
(type=int)

`kickReason:` the reason for the kick
(type=string)

Return Value

the errorcode
(type=int)

setChannelVariableAsInt(*serverConnectionHandlerID*, *channelID*, *flag*, *value*)

Sets a channel variable to a new int value.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel, pass 0 to set a new channel's variables *(type=int)*
`flag:`	the flag to set *(type=int)*
`value:`	the new value *(type=int)*

Return Value

the errorcode

(type=int)

guiConnect(*connectTab*, *serverLabel*, *serverAddress*, *serverPassword*, *nickname*, *channel*, *channelPassword*, *captureProfile*, *playbackProfile*, *hotkeyProfile*, *userIdentity*, *oneTimeKey*, *phoneticName*)

Connects to a server and displays it as tab in the client.

Parameters

`connectTab:` defines, which tab will be used, see ts3defines.PluginConnectTab
(type=int)

`serverLabel:` the label of the connection
(type=string)

`serverAddress:` host or ip address
(type=string)

`serverPassword:` password to the server, pass an empty string if the server is not password protected
(type=string)

`nickname:` the user's nickname
(type=string)

`channel:` complete path to the channel to connect to
(type=string)

`channelPassword:` password to the channel, pass an empty string if the channel is not password protected
(type=string)

`captureProfile:` the name of the capture profile to use
(type=string)

`playbackProfile:` the name of the playback profile to use
(type=string)

`hotkeyProfile:` the name of the hotkey profile to use
(type=string)

`userIdentity:` identity to use
(type=string)

`oneTimeKey:` privilege key to use on connect
(type=string)

`phoneticName:` phonetic nickname
(type=string)

Return Value

a tuple, containing the errorcode and the ID of the created serverconnection handler
(type=tuple (int, int))

bandelall(*serverConnectionHandlerID*, *returnCode*)

Deletes all bans on a server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

bandel(*serverConnectionHandlerID*, *banID*, *returnCode*)

Deletes a ban.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`banID:` the ID of the ban
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestClientPoke(*serverConnectionHandlerID*, *clientID*, *message*, *returnCode*)

Pokes a client with a given message.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`message:` the message
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getPreProcessorInfoValue(*serverConnectionHandlerID*, *ident*)

Querie a sound preprocessor flag and returns it as string.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`ident:` the flag to be queried
(type=string)

Return Value

A tuple, containing the errorcode and the value of the queried flag
(type=tuple (int, string))

requestConnectionInfo(*serverConnectionHandlerID*, *clientID*, *returnCode*)

Requests the connection info of a client. The event onConnectionInfoEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getPlaybackModeList()

Queries all available playback modes.

Return Value

A tuple, containing the errorcode and the list of modes
(type=tuple (int, [string]))

showHotkeySetup()

requestMessageGet(*serverConnectionHandlerID*, *messageID*, *returnCode*)

Requests an offline message defined by its ID. The event onMessageGetEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`messageID:` the ID of the message
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getCaptureDeviceList(*modeID*)

Queries all available capture devices.

Parameters

`modeID:` Defines the capture mode to use.
(type=string)

Return Value

A tuple, containing the errorcode and the list of capture devices as tuple (devicename, deviceid)
(type=tuple (int, [(string, string)]))

requestChannelDelete(*serverConnectionHandlerID*, *channelID*, *force*, *returnCode*)

Deletes a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelID:` the ID of the channel to delete

(type=int)

`force:` if set to 1 (or True), the channel will be deleted even when it is not empty

(type=int or bool)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

getDirectories(*path*, *maxLen*=256)

//FIXME: wäh?

Parameters

`path:` *(type=)*

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

the resulting path

(type=string)

sendFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *overwrite*, *resume*, *sourceDirectory*, *returnCode*)

Starts a fileupload to the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelID:` the ID of the channel in which the file will be placed in

(type=int)

`channelPW:` the password of the channel, pass an empty string if the channel is not password protected

(type=string)

`file:` the complete path to the file in the channel

(type=string)

`overwrite:` if set to 1 (or True) and a file with that name already exists will be overwritten

(type=int or bool)

`resume:` if set to 1 (or True), a previously started filetransfer can be resumed

(type=int or bool)

`sourceDirectory:` the directory on the system, where the original file is placed in

(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

a tuple, containing the errorcode and the ID of the filetransfer

(type=tuple (int, int))

getTransferFileSizeDone(*transferID*)

Returns the already downloaded size (in Bytes) of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer

(type=int)

Return Value

a tuple, containing the errorcode and the size

(type=tuple (int, int))

sendPluginCommand(*serverConnectionHandlerID*, *command*, *targetMode*, *targetIDs*, *returnCode*)

Sends a plugin command to other users.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`command:` the command string

(type=string)

`targetMode:` specifies, to whom the command will be send, see ts3defines.PluginTargetMode

(type=int)

`targetIDs:` a list of client IDs, only needed if targetMode == ts3defines.PluginTargetMode.PluginCommandTarget_CLIENT

(type=list [int])

getErrorMessage(*errorCode*)

Queries a printable error string for a specific error code.

Parameters

`errorCode:` The error code returned from all Client Lib functions

(type=int)

Return Value

A tuple, containing the errorcode and the resulting string

(type=tuple (int, string))

requestPermissionOverview(*serverConnectionHandlerID*, *clientDBID*, *channelID*, *returnCode*)

Requests the permission overview of a user in a channel. The events onPermissionOverviewEvent and onPermissionOverviewFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientDBID:`	the database ID of the user *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestClientPermList(*serverConnectionHandlerID*, *clientDatabaseID*, *returnCode*)

Requests the list of permissions assigned to a user. The events

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

requestChannelUnsubscribe(*serverConnectionHandlerID*, *channelIDArray*, *returnCode*)

Unsubscribes from a list channels.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelIDArray:` a list of channel IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

setChannelVariableAsUInt64(*serverConnectionHandlerID*, *channelID*, *flag*, *value*)

Sets a channel variable to a new unsigned long long int value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel, pass 0 to set a new channel's variables
(type=int)

`flag:` the flag to set
(type=int)

`value:` the new value
(type=int)

Return Value

the errorcode
(type=int)

requestChannelClientPermList(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *returnCode*)

Requests the list of permissions of a user in a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

systemset3DSettings(*serverConnectionHandlerID*, *distanceFactor*, *rolloffScale*)

Adjust 3D sound system settings.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`distanceFactor:`	relative distance factor. Default is 1.0 = 1 meter *(type=float)*
`rolloffScale:`	Scaling factor for 3D sound rolloff. Defines how fast sound volume will attenuate. As higher the value, as faster the sound is toned with increasing distance. *(type=float)*

Return Value

the errorcode

(type=int)

requestClientDBIDfromUID(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *returnCode*)

Requests the database ID of a client defined by the UID. The event onClientDBIDfromUIDEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientUniqueIdentifier:`	the UID of the client *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestSendPrivateTextMsg(*serverConnectionHandlerID*, *message*, *targetClientID*, *returnCode*)

Sends a private text message to a client.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`message:`	the message to send *(type=string)*
`targetClientID:`	the ID of the client to send the message to *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelGroupAddPerm(*serverConnectionHandlerID*, *channelGroupID*, *continueonerror*, *permissionIDArray*, *permissionValueArray*, *returnCode*)

Adds a list of permissions to a channelgroup.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelGroupID:`	the ID of the channelgroup *(type=int)*
`continueonerror:`	if set to True, if an error with a permission occurs, the other permissions will even though be handled *(type=bool)*
`permissionIDArray:`	a list of permission IDs *(type=list [int])*
`permissionValueArray:`	list of permission values, in order of the permissions in permissionIDArray *(type=list [int])*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelUnsubscribeAll(*serverConnectionHandlerID*, *returnCode*)

Unsubscribes from all channels on the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getPluginPath(*maxLen*=256)

Returns the ts3 plugin path.

Parameters

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256
(type=int)

Return Value

the plugin path
(type=string)

requestInfoUpdate(*serverConnectionHandlerID*, *itemType*, *itemID*)

Requests to update the info data.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`itemType:` specifies, which info data update is requested, see ts3defines.PluginItemType
(type=int)

`itemID:` the ID of the item //FIXME: implement!
(type=int)

Return Value

the errorcode
(type=int)

printMessage(*serverConnectionHandlerID*, *message*, *messageTarget*)

Prints a message to a specific client chat tab.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`message:` the message to print
(type=string)

`messageTarget:` the target to send the message, see ts3defines.PluginMessageTarget
(type=int)

privilegeKeyUse(*serverConnectionHandlerID*, *tokenKey*, *returnCode*)

Uses a privilege key as the current client of the serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`tokenKey:` the token
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getCurrentCaptureMode(*serverConnectionHandlerID*)

Queries the current capture mode on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection
(type=int)

Return Value

A tuple, containing the errorcode and the current capture mode
(type=tuple (int, string))

requestSendServerTextMsg(*serverConnectionHandlerID*, *message*, *returnCode*)

Sends a text message to all clients on the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`message:` the message to send
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

spawnNewServerConnectionHandler(*port*)

Creates a new server connection handler and receive its ID.

Parameters

`port:` Port the client should bind on. Specify zero to let the operating system chose any free port
(type=int)

Return Value

A tuple, containig the errorcode and the resulting ID
(type=tuple (int, int))

pauseWaveFileHandle(*serverConnectionHandlerID*, *waveHandle*, *pause*)

Pauses a wavefile sound previously started with playWaveFileHandle.

Parameters

`serverConnectionHandlerID:` the ID of the serverConnection the sound is played on

(type=int)

`waveHandle:` the handle returned by playWaveFileHandle

(type=int)

`pause:` if set to 1 (or True), the sound will pause, 0 (or False) will unpause the sound

(type=int or bool)

Return Value

the errorcode

(type=int)

getTransferFileSize(*transferID*)

Returns the total filesize (in Bytes) of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer

(type=int)

Return Value

a tuple, containing the errorcode and the filesize

(type=tuple (int, int))

channelset3DAttributes(*serverConnectionHandlerID*, *clientID*, *position*)

Adjusts a clients position and velocity in 3D space.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientID:`	the ID of the client to adjust *(type=int)*
`position:`	a tuple defining the position of the clientID *(type=tuple (float, float, float))*

Return Value

the errorcode

(type=int)

requestClientMove(*serverConnectionHandlerID*, *clientID*, *newChannelID*, *password*, *returnCode*)

Moves a client to a different channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`clientID:` the ID of the client to be moved

(type=int)

`newChannelID:` the ID of the channel moving the client to

(type=int)

`password:` password of the channel, leave empty if channel is not password protected

(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

requestChannelGroupDelPerm(*serverConnectionHandlerID*, *channelGroupID*, *continueOnError*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions from a channelgroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelGroupID:` the ID of the channelgroup
(type=int)

`continueOnError:` if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled
(type=int or bool)

`permissionIDArray:` a list of permission IDs
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

setChannelVariableAsString(*serverConnectionHandlerID*, *channelID*, *flag*, *value*)

Sets a channel variable to a new string value.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`channelID:` the ID of the channel, pass 0 to set a new channel's variables
(type=int)

`flag:` the flag to set
(type=int)

`value:` the new value
(type=int)

Return Value

the errorcode
(type=int)

getConnectionStatus(*serverConnectionHandlerID*)

Returns the current connection status of a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

a tuple, containing the errorcode and the connection status
(type=tuple (int, int))

requestClientNamefromDBID(*serverConnectionHandlerID*, *clientDatabaseID*, *returnCode*)

Requests the name of a client defined by the database ID. The event onClientNamefromDBIDEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientDatabaseID:`	the database ID of the client *(type=int)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestComplainDel(*serverConnectionHandlerID*, *targetClientDatabaseID*, *fromClientDatabaseID*, *returnCode*)

Deletes a complain to a user by a different user.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`targetClientDatabaseID:` the database ID of the complained user
(type=int)

`fromClientDatabaseID:` the database ID of the complaining user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getClientLibVersion()

Returns the clientlib's version as string.

Return Value

A tuple, containing the errorcode and the result
(type=tuple (int, string))

clientPropertyStringToFlag(*clientPropertyString*)

//FIXME:

Parameters

`clientPropertyString:` *(type=string)*

Return Value

a tuple, containing the errorcode and
(type=tuple (int, int))

setPluginMenuEnabled(*menuID*, *enabled*)

Enables or disables a menuitem. The menuID must be the global id, not the local id plugin developers set in menuItems. Retrieve it with PluginHost.globalMenuID.

Parameters

`menuID:` global id of the menuitem

(type=int)

`enabled:` set to True to enable it, False otherwise

(type=bool)

requestClientEditDescription(*serverConnectionHandlerID*, *clientID*, *clientDescription*, *returnCode*)

Sets the description of a client.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`clientID:` the ID of the client

(type=int)

`clientDescription:` the description to set

(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

destroyServerConnectionHandler(*serverConnectionHandlerID*)

Destroys a server connection handler.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

The errorcode
(type=int)

requestServerGroupAddClient(*serverConnectionHandlerID*, *serverGroupID*, *clientDatabaseID*, *returnCode*)

Adds a user defined by his database ID to a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getDefaultPlayBackMode()

Queries the default playback mode.

Return Value

A tuple, containing the errorcode and the default playback mode
(type=tuple (int, string))

activateCaptureDevice(*serverConnectionHandlerID*)

Activates the capture device on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

the errorcode
(type=int)

getConnectionVariableAsString(*serverConnectionHandlerID*, *clientID*, *flag*)

Returns a client's connection variable as string variable.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientID:` the ID of the client
(type=int)

Return Value

a tuple, containing the errorcode and the value of the flag
(type=tuple (int, string))

openPlaybackDevice(*serverConnectionHandlerID*, *modeID*, *playbackDevice*)

Opens a playback device on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection on which the playback device should be initialized on
(type=int)

`modeID:` the playback mode to use
(type=string)

`playbackDevice:` the id of the playback device
(type=string)

Return Value

the errorcode
(type=int)

requestHotkeyInputDialog(*keyword*, *isDown*, *qParentWindow*)

Parameters

`keyword:` *(type=)*

`isDown:` *(type=)*

`qParentWindow:` *(type=)*

requestServerTemporaryPasswordDel(*serverConnectionHandlerID*, *password*, *returnCode*)

Deletes an existing temporary password.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`password:` the password to delete
(type=string)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

(type=)

getClientNeededPermission(*serverConnectionHandlerID*, *permissionName*)

//FIXME: wäh?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`permissionName:` name of the permission
(type=string)

Return Value

a tuple, containing the errorcode and
(type=tuple (int, int))

requestServerGroupAddPerm(*serverConnectionHandlerID*, *serverGroupID*, *continueonerror*, *permissionIDArray*, *permissionValueArray*, *permissionNegatedArray*, *permissionSkipArray*, *returnCode*)

Adds a list of permissions to a servergroup.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`continueonerror:` if set to 1 (or True), if an error with a permission occurs, the other permissions will even though be handled
(type=int or bool)

`permissionIDArray:` list of permission IDs
(type=list [int])

`permissionValueArray:` list of permission values, in order of the permissions in permissionIDArray
(type=list [int])

`permissionNegatedArray:` list of permission negated values, in order of the permissions in permissionIDArray
(type=list [int])

`permissionSkipArray:` list of permission skip values, in order of the permissions in permissionIDArray
(type=list [int])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

startVoiceRecording(*serverConnectionHandlerID*)

Starts voice recording on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

Return Value

the errorcode

(type=int)

requestServerGroupList(*serverConnectionHandlerID*, *returnCode*)

Requests the list of servergroups. The events onServerGroupListEvent and onServerGroupListFinishedEvent will be triggered.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

requestFileInfo(*serverConnectionHandlerID*, *channelID*, *channelPW*, *file*, *returnCode*)

Requests the info to a file in a channel. The event onFileInfoEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`channelPW:`	the password of the channel, pass an empty string if the channel is not password protected *(type=string)*
`file:`	the complete path to the file *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

getPermissionIDByName(*serverConnectionHandlerID*, *permissionName*)

Returns the ID of a permission defined by its name.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`permissionName:`	name of the permission *(type=string)*

Return Value

a tuple, containing the errorcode and the ID of the permission

(type=tuple (int, int))

isWhispering(*serverConnectionHandlerID*, *clientID*)

Returns the status of a client whether he is currently whispering. //FIXME: only to me?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

Return Value

a tuple, containing the errorcode and the status
(type=tuple (int, int or bool))

requestClientAddPerm(*serverConnectionHandlerID*, *clientDatabaseID*, *permissionIDArray*, *permissionValueArray*, *permissionSkipArray*, *returnCode*)

Adds a list of permissions to a user.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`clientDatabaseID:` the database ID of the user
(type=int)

`permissionIDArray:` a list of permission IDs
(type=list [int])

`permissionValueArray:` list of permission values, in order of the permissions in permissionIDArray
(type=list [int])

`permissionSkipArray:` list of permission skip values, in order of the permissions in permissionIDArray
(type=list [int])

Return Value

the errorcode
(type=int)

requestDeleteFile(*serverConnectionHandlerID*, *channelID*, *channelPW*, *files*, *returnCode*)

Deletes a list of files in a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelID:` the ID of the channel

(type=int)

`channelPW:` the password of the channel, pass an empty string if channel is not password protected

(type=string)

`files:` a list of complete pathes of the file to delete

(type=list [string])

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

requestSendChannelTextMsg(*serverConnectionHandlerID*, *message*, *targetChannelID*, *returnCode*)

Sends a text message to all clients in a channel.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`message:` the message to send
(type=string)

`targetChannelID:` the ID of the channel
(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getServerConnectionHandlerList()

Returns a list of serverconnection handlers.

Return Value

a tuple, containing the errorcode and the list of serverconnection handler IDs
(type=tuple (int, [int]))

set3DWaveAttributes(*serverConnectionHandlerID*, *waveHandle*, *position*)

Positions a wave file that was opened previously with playWaveFileHandle in 3D space.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`waveHandle:` handle of the played wavefile sound returned by playWaveFileHandle
(type=int)

`position:` A tuple defining the 3D position of the sound
(type=tuple (float, float, float))

Return Value

the errorcod
(type=int)

getCurrentPlaybackDeviceName(*serverConnectionHandlerID*)

Queries the current playback device's name on a serverconnection.

Parameters

`serverConnectionHandlerID:` ID of the serverconnection
(type=int)

Return Value

A tuple, containing the errorcode, the playback device's name and the status, if it's default
(type=tuple (int, string, int))

getTransferRunTime(*transferID*)

Returns the runtime of a filetransfer.

Parameters

`transferID:` the ID of the filetransfer
(type=int)

Return Value

a tuple, containing the errorcode and the runtime //FIXME: seconds? miliseconds?
(type=tuple (int, int))

getPlaybackDeviceList(*modeID*)

Queries all available playback devices.

Parameters

`modeID:` Defines the playback mode to use.

(type=string)

Return Value

A tuple, containing the errorcode and the list of playback devices as tuple (devicename, deviceid)

(type=tuple (int, [(string, string)]))

playWaveFile(*serverConnectionHandlerID*, *path*)

Plays a wavefile sound on a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`path:` the path to the wavefile on the system

(type=string)

Return Value

the errorcode

(type=int)

banadd(*serverConnectionHandlerID*, *ipRegExp*, *nameRegexp*, *uniqueIdentity*, *timeInSeconds*, *banReason*, *returnCode*)

Adds a new ban.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`ipRegExp:`	regular expression to match IPs, pass an empty string to ignore IPs *(type=string)*
`nameRegexp:`	regular expression to match client nicknames, pass an empty string to ignore nicknames *(type=string)*
`uniqueIdentity:`	client UID to ban, pass an empty string to ignore UIDs *(type=string)*
`timeInSeconds:`	the time, the client should be banned for, pass 0 to add a permanent ban *(type=int)*
`banReason:`	the reason for the ban *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestBanList(*serverConnectionHandlerID*, *returnCode*)

Requests the banlist on a server. The event onBanListEvent will be triggered. //FIXME: ts3plugin_onBanListFinishedEvent is missing?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

getServerConnectInfo(*serverConnectionHandlerID*, *maxLen*=`256`)

Returns the connect info (host, port and password) of a serverconnection.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

a tuple, containing the errorcode, the host, the port and the password of the serverconnection

(type=tuple (int, string, int, string))

requestClientNamefromUID(*serverConnectionHandlerID*, *clientUniqueIdentifier*, *returnCode*)

Requests the name of a client defined by the UID. The event onClientNamefromUIDEvent will be triggered.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`clientUniqueIdentifier:`	the UID of the client *(type=string)*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

requestChannelClientDelPerm(*serverConnectionHandlerID*, *channelID*, *clientDatabaseID*, *permissionIDArray*, *returnCode*)

Deletes a list of permissions of a user in a channel.

Parameters

`serverConnectionHandlerID:`	the ID of the serverconnection *(type=int)*
`channelID:`	the ID of the channel *(type=int)*
`clientDatabaseID:`	the database ID of the user *(type=int)*
`permissionIDArray:`	a list of permission IDs *(type=list [int])*
`returnCode:`	returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional. *(type=string)*

Return Value

the errorcode

(type=int)

flushChannelCreation(*serverConnectionHandlerID*, *channelParentID*, *returnCode*)

Flushes the channel creation made by the setChannelVariable-functions to the server.

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection

(type=int)

`channelParentID:` the ID of the parent channel of the new channel

(type=int)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.

(type=string)

Return Value

the errorcode

(type=int)

createReturnCode(*maxLen*=`128`)

Creates a returnCode which can be passed to the other functions and will be passed to the event onServerErrorEvent.

Parameters

`maxLen:` length of the buffer, passed to the clientlib to store the path to, default value is 256

(type=int)

Return Value

the created returnCode

(type=string)

requestServerGroupClientList(*serverConnectionHandlerID*, *serverGroupID*, *withNames*, *returnCode*)

Requests the list of clients assigned to a servergroup. The event onServerGroupClientListEvent will be triggered. //FIXME: ts3plugin_onServerGroupClientListFinishedEvent is missing?

Parameters

`serverConnectionHandlerID:` the ID of the serverconnection
(type=int)

`serverGroupID:` the ID of the servergroup
(type=int)

`withNames:` if set to 1 (or True), the event will contain clientNameIdentifier of the user instead of an empty string //FIXME: right?
(type=int or bool)

`returnCode:` returnCode passed to onServerErrorEvent or onServerPermissionErrorEvent. Optional.
(type=string)

Return Value

the errorcode
(type=int)

getClientLibVersionNumber()

Returns the clientlib's version number

Return Value

A tuple, containing the errorcode and the result
(type=tuple (int, int))

getDefaultCaptureMode()

Queries the default capture mode.

Return Value

A tuple, containing the errorcode and the default capture mode
(type=tuple (int, string))

Index