**Teleperformance**

***Encuesta application***

**MAIN PROCESS**

get ARI & MQ handlers

while( true) {

if(event Received ) {

if event = StasisStart

Start New Thread();

Put Thread in HashMap(), associated with channel;

Else if event = StasisEnd

getThread handle from HashMap;

Else

Send event to Thread;

}

Get Actions to execute from Thread

Execute Actions; // WE STILL DO NOT KNOW HOW THIS WILL BE PERFORMED !!!!!

}

**DETAILED METHODS (some of them, not all, are detailed here)**

----------------------------------------------------------------------------------------------------------------------------

ari = getAriConnection(lastAriEventTimer) {

if exists

if lastAriEventTimer.check() // if more than 5 minutes since last AriEvent

lastAriEventTimer.reset();

perform a keep alive operation (Get info)

if keep alive fails, cleanup ari connection, get connection configuration parameters from properties, open an ari connection and return the new connection

return current connection (singleton)

else

lastAriEventTimer.reset();

get connection configuration parameters from properties file, open an ari connection and return the new connection

// Note: if IP is not correct or there are network problems AriFactory.nettyHttp() is blocking (never returns) ----> This needs to be fixed !!!

}

----------------------------------------------------------------------------------------------------------------------------

checkStateMachineTimers () {

while (browse smTimers list)

if timer has expired

remove timer from list;

return channel;

}

----------------------------------------------------------------------------------------------------------------------------

processEvent(smEvent) {

ariActionList = new AriActionList();

if smEvent.type = ChannelDtmfReceived

if sm.State(sm.CurrentState).bargeIn=true

add stopPlayback to ariActionList;

else if smEvent.type = StasisStart

index = currentState = 0;

return sm.State(currentState).ariAction list;

else if smEvent.type = StasisEnd

storeSurveyReportToDb(); // where do we get the values from ?????

return null;

else if smEvent.type = PlaybackFinished

timeoutToCheck = sm.State(sm.CurrentState).timeout

if timeoutToCheck != null and > 0

ariAction = start timer for smEvent.channel on timeoutToCheck;

return ariAction list;

else

smEvent.type = NoInput;

// PENDING HERE !!!!!!!!!!!!!!!!!!!

on current state of channel {

browse transitionEvent list with smEvent;

if found {

if threshold <= 0 then, update currentState index;

else decrement threshold;

}

else // NoMatch assumed {

}

}

returnAction = State(index).Action;

return returnAction;

}

**STATE MACHINE GENERIC DTOs**

**SmEvent**

Event data passed to the State Machine for processing

|  |  |  |
| --- | --- | --- |
| DTO Class: SmEvent | Type | Possible values & Comments |
| channel | String | Channel id from ARI |
| type | String | StasisStart/null  PlaybackFinished/playbackId  ChannelDtmfReceived/DtmfValue  StasisEnd or ChannelHangupRequest/null |
| value | String |

**SmAction**

A list of SmAction is returned from the State Machine to the main while loop for processing

|  |  |  |
| --- | --- | --- |
| DTO Class: SmAction | Type | Possible values & Comments |
| channel | String |  |
| type | String | PLAY / Instance of SmActionPlayValue  STOP\_PLAY / Instance of String (playBackId)  HANGUP / null  REPORT\_TO\_DB / Instance of SurveyReport  RECORD\_CDR / ?????????????? |
| value | Object |

|  |  |  |
| --- | --- | --- |
| DTO Class: SmActionPlayValue | Type | Possible values & Comments |
| audioFileName | String |  |
| playBackId | String |  |

**CallDetailRecord**

Holds the report data to be sent to the database at the end of the call

|  |  |  |
| --- | --- | --- |
| DTO Class: CallDetailRecord | Type | Possible values & Comments |
| callData | CallData (Obj) |  |
|  |  | We need to add more info here |

**SURVEY APPLICATION SPECIFIC DTOs**

**SurveyTemplate**

Holds the information for each survey and is loaded from the survey configuration file. See Survey Voice User Interface (VUI) Specifications for more details.

|  |  |  |
| --- | --- | --- |
| DTO Class: SurveyTemplate | Type | Possible values & Comments |
| surveyCode | String |  |
| timeStartDay | String | Default = 08:00:00 |
| timeStartAfternoon | String | Default = 14:00:00 |
| timeStartNight | String | Default = 19:00:00 |
| audioWelcomeDay | String |  |
| audioWelcomeAfternoon | String |  |
| audioWelcomeNight | String |  |
| audioWelcome | String |  |
| audioWelcomeBargeIn | Boolean | Default = true |
| audioGoodbye | String |  |
| questions (list) | Question (Obj) |  |

|  |  |  |
| --- | --- | --- |
| DTO Class: Question | Type | Possible values & Comments |
| number | Integer |  |
| audioIntro | String |  |
| audioIntroBargeIn | Boolean | Default = true |
| audio | String |  |
| audioBargeIn | Boolean | Default = true |
| noMatchRetries | Integer | Default = 0 |
| noInputRetries | Integer | Default = 0 |
| uniqueCounter | Boolean | Default = true |
| transitions (list) | QuestionTransitions (Obj) |  |

|  |  |  |
| --- | --- | --- |
| DTO Class: QuestionTransitions | Type | Possible values & Comments |
| event | String |  |
| action | String |  |

**SurveyReport**

Holds the report data to be sent to the database at the end of the call

|  |  |  |
| --- | --- | --- |
| DTO Class: SurveyReport | Type | Possible values & Comments |
| surveyCode | String | from Config File |
| timeStart | Calendar |  |
| timeEnd | Calendar |  |
| ani | String | From Call Data |
| dnis | String | From Call Data |
| agentId | String | From Call Data |
| resp1 | String |  |
| resp2 | String |  |
| resp3 | String |  |
| resp4 | String |  |
| resp5 | String |  |
| resp6 | String |  |
| resp7 | String |  |
| resp8 | String |  |
| resp9 | String |  |
| resp10 | String |  |