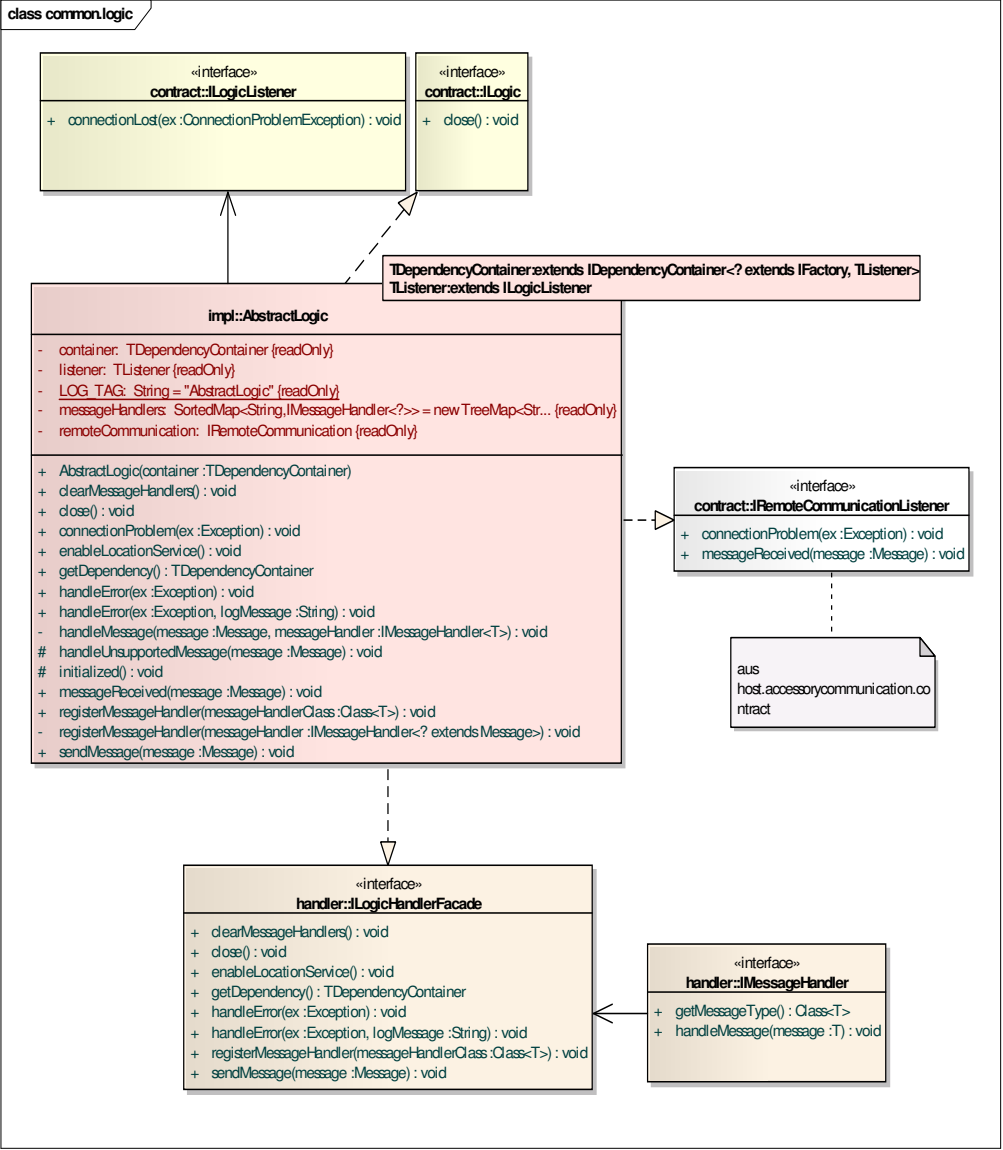
class common.logic
«interface»
contract::ILogicListener
+ connectionLost(ex :ConnectionProblemException) : void
«interface»
contract::ILogic
+ close() : void
TDependencyContainer:extends IDependencyContainer<? extends IFactory, TListener>
TListener:extends ILogicListener
impl::AbstractLogic
- container: TDependencyContainer {readOnly}
- listener: TListener {readOnly}
- LOG_TAG: String = "AbstractLogic" {readOnly}
- messageHandlers: SortedMap<String,IMessageHandler<?>> = new TreeMap<Str... {readOnly}
- remoteCommunication: IRemoteCommunication {readOnly}
+ AbstractLogic(container :TDependencyContainer)
+ clearMessageHandlers() : void
+ close() : void
+ connectionProblem(ex :Exception) : void
+ enableLocationService() : void
+ getDependency() : TDependencyContainer
+ handleError(ex :Exception) : void
+ handleError(ex :Exception, logMessage :String) : void
- handleMessage(message :Message, messageHandler :IMessageHandler<T>) : void
handleUnsupportedMessage(message :Message) : void
initialized() : void
+ messageReceived(message :Message) : void
+ registerMessageHandler(messageHandlerClass :Class<T>) : void
- registerMessageHandler(messageHandler :IMessageHandler<? extends Message>) : void
+ sendMessage(message :Message) : void
«interface»
contract::IRemoteCommunicationListener
+ connectionProblem(ex :Exception) : void
+ messageReceived(message :Message) : void
aus
host.accessorycommunication.co
ntract
«interface»
handler::ILogicHandlerFacade
+ clearMessageHandlers() : void
+ close() : void
+ enableLocationService() : void
+ getDependency() : TDependencyContainer
+ handleError(ex :Exception) : void
+ handleError(ex :Exception, logMessage :String) : void
+ registerMessageHandler(messageHandlerClass :Class<T>) : void
+ sendMessage(message :Message) : void
«interface»
handler::IMessageHandler
+ getMessageType() : Class<T>
+ handleMessage(message :T) : void