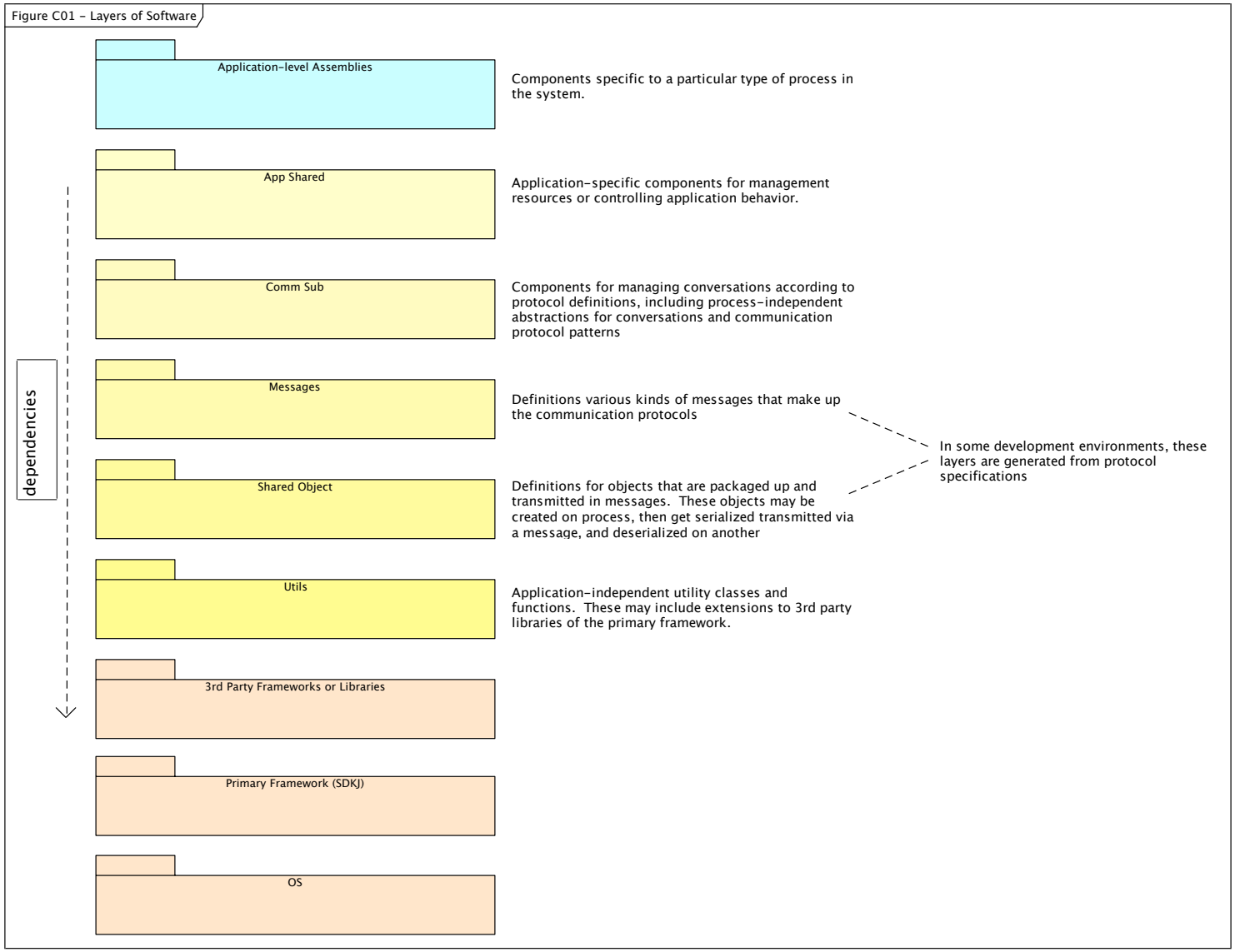
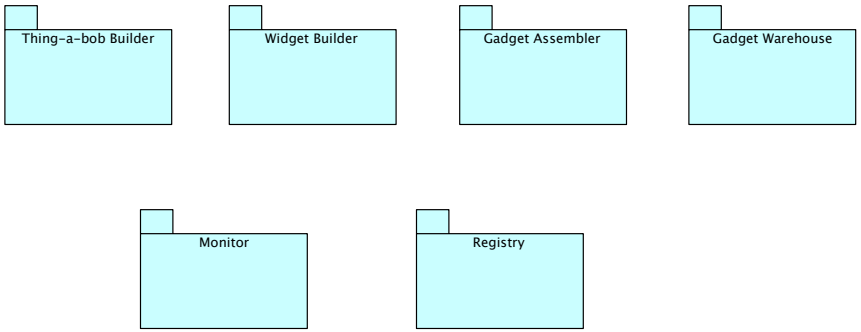
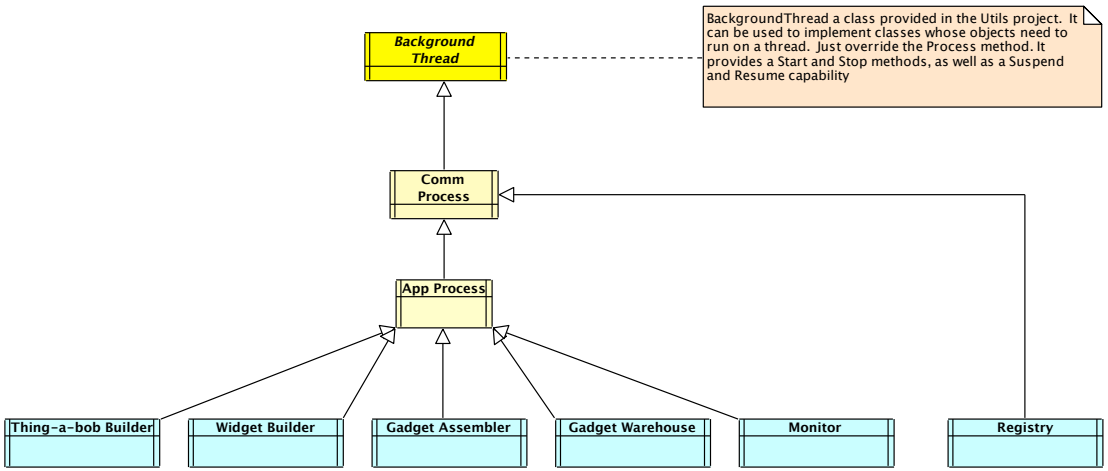
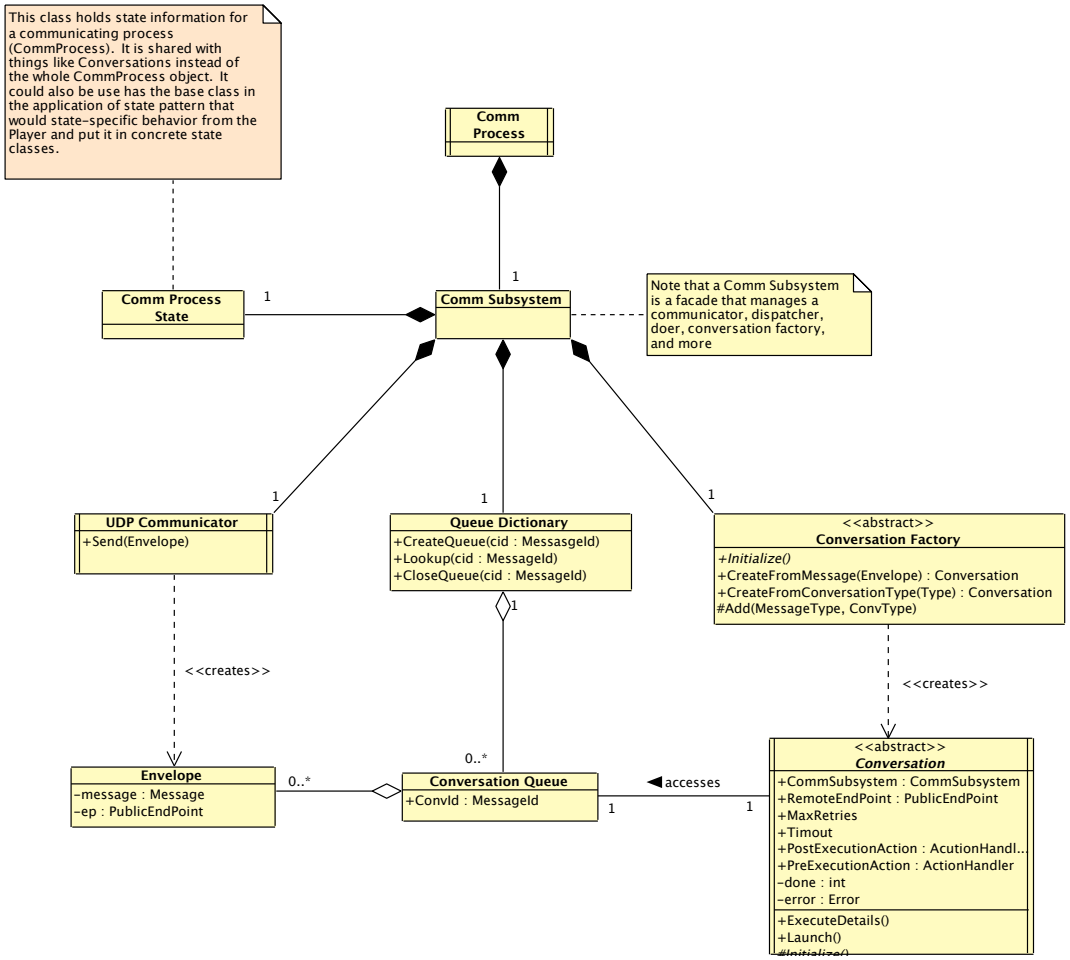
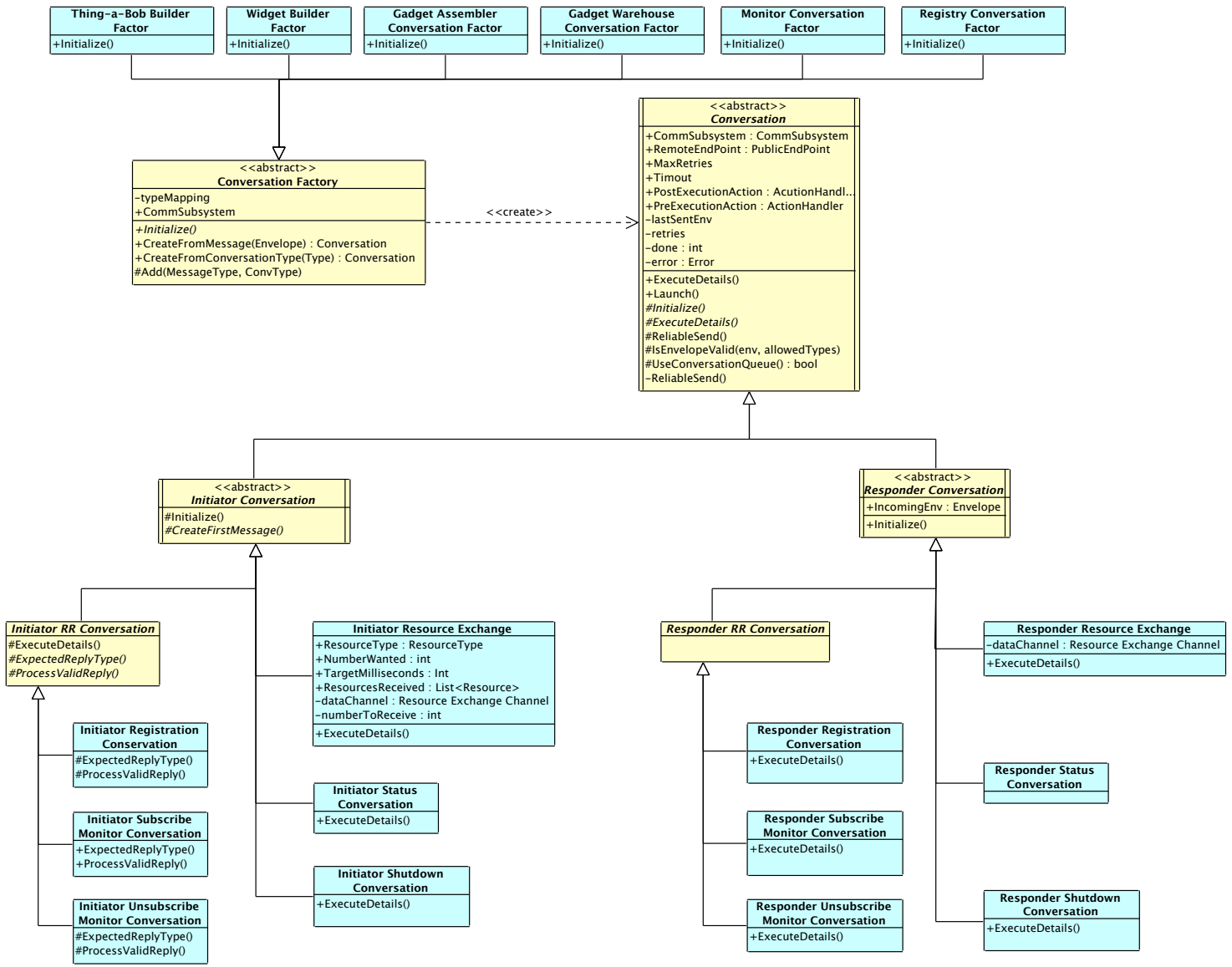


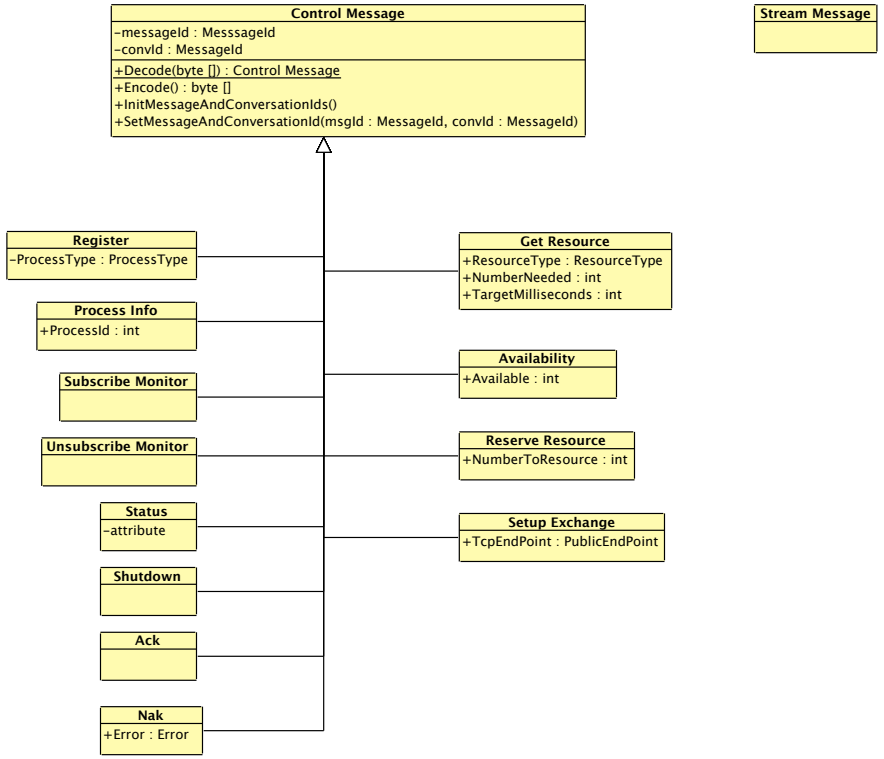
Figure C01 - Layers of Software
Application-level Assemblies
Components specific to a particular type of process in the system.
App Shared
Application-specific components for management resources or controlling application behavior.
Comm Sub
Components for managing conversations according to protocol definitions, including process-independent abstractions for conversations and communication protocol patterns
Messages
Definitions various kinds of messages that make up the communication protocols
Shared Object
Definitions for objects that are packaged up and transmitted in messages. These objects may be created on process, then get serialized transmitted via a message, and deserialized on another
In some development environments, these layers are generated from protocol specifications
Utils
Application-independent utility classes and functions. These may include extensions to 3rd party libraries of the primary framework.
3rd Party Frameworks or Libraries
Primary Framework (SDKJ)
OS
dependencies

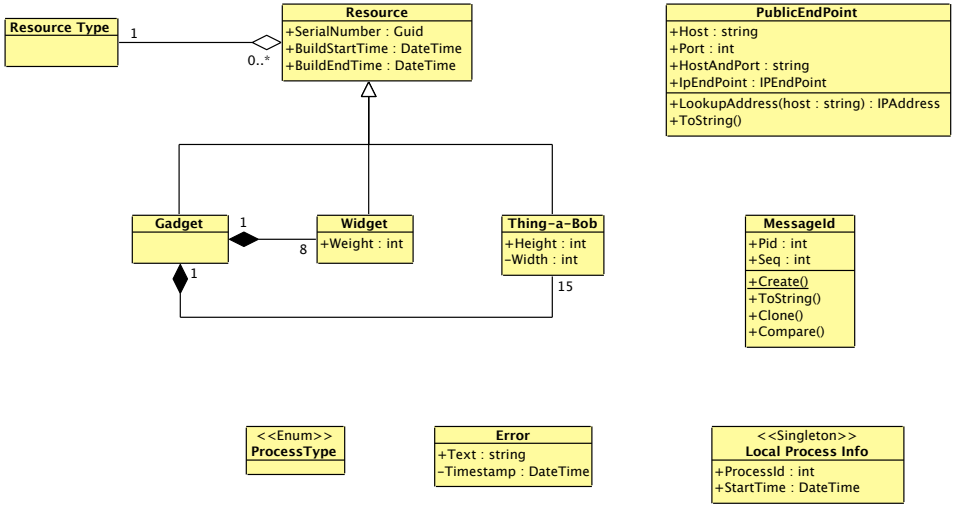

Thing-a-bob Builder
Widget Builder
Gadget Assembler
Gadget Warehouse
Monitor
Registry

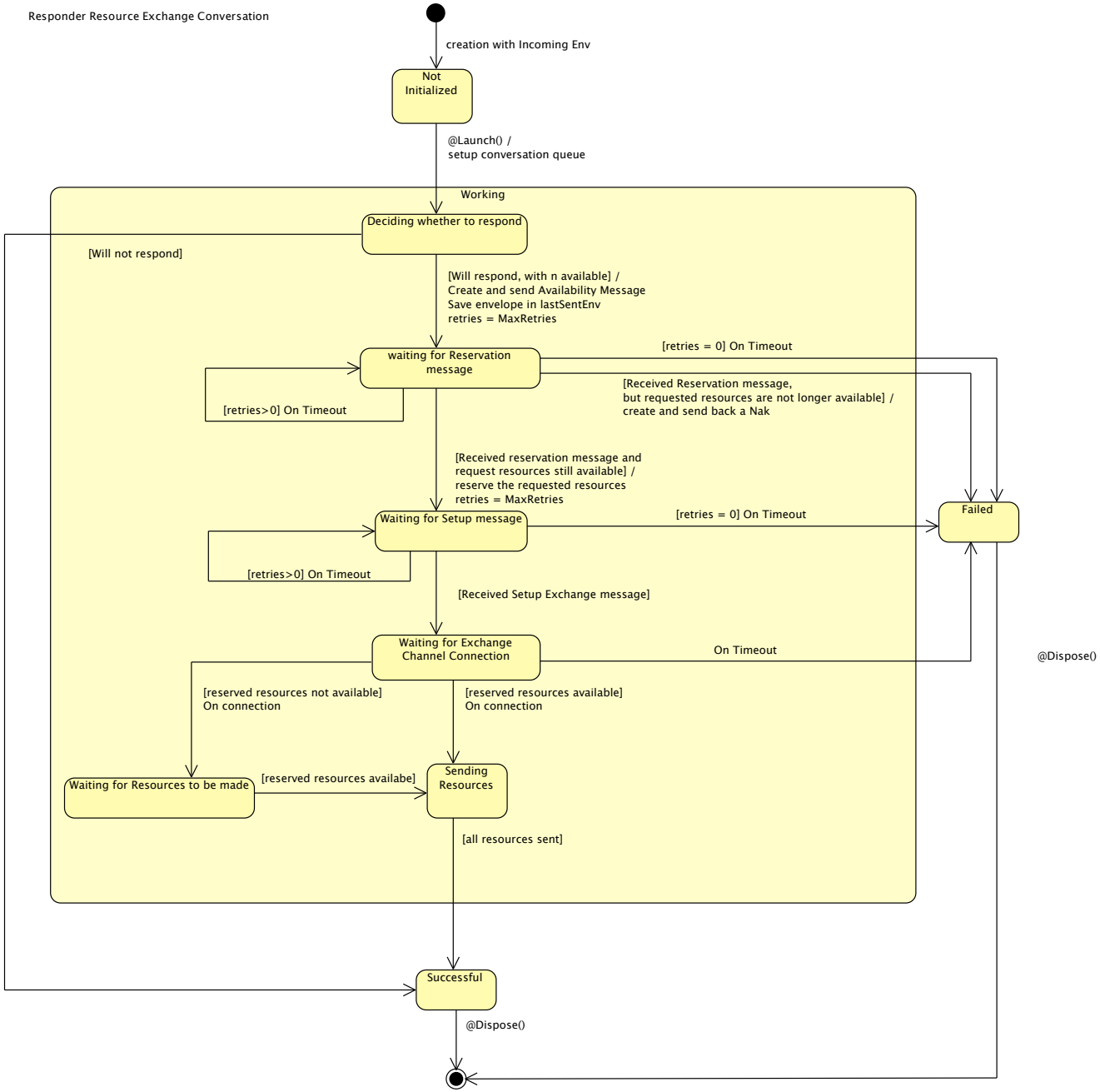

Background Thread
BackgroundThread a class provided in the Utils project. It can be used to implement classes whose objects need to run on a thread. Just override the Process method. It provides a Start and Stop methods, as well as a Suspend and Resume capability
Comm Process
App Process
Thing-a-bob Builder
Widget Builder
Gadget Assembler
Gadget Warehouse
Monitor
Registry

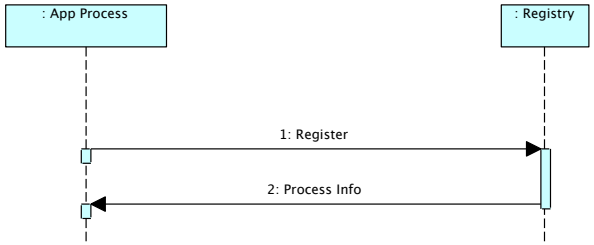

This class holds state information for a communicating process (CommProcess). It is shared with things like Conversations instead of the whole CommProcess object. It could also be use has the base class in the application of state pattern that would state-specific behavior from the Player and put it in concrete state classes.
Comm Process
Comm Process State
1
Comm Subsystem
1
Note that a Comm Subsystem is a facade that manages a communicator, dispatcher, doer, conversation factory, and more
1
1
1
UDP Communicator
+Send(Envelope)
Queue Dictionary
+CreateQueue(cid : MessasgeId)
+Lookup(cid : MessageId)
+CloseQueue(cid : MessageId)
<<abstract>>
Conversation Factory
+Initialize()
+CreateFromMessage(Envelope) : Conversation
+CreateFromConversationType(Type) : Conversation
#Add(MessageType, ConvType)
<<creates>>
1
<<creates>>
Envelope
-message : Message
-ep : PublicEndPoint
0..*
0..*
Conversation Queue
+ConvId : MessageId
accesses
1
1
<<abstract>>
Conversation
+CommSubsystem : CommSubsystem
+RemoteEndPoint : PublicEndPoint
+MaxRetries
+Timout
+PostExecutionAction : AcutionHandl...
+PreExecutionAction : ActionHandler
-done : int
-error : Error
+ExecuteDetails()
+Launch()

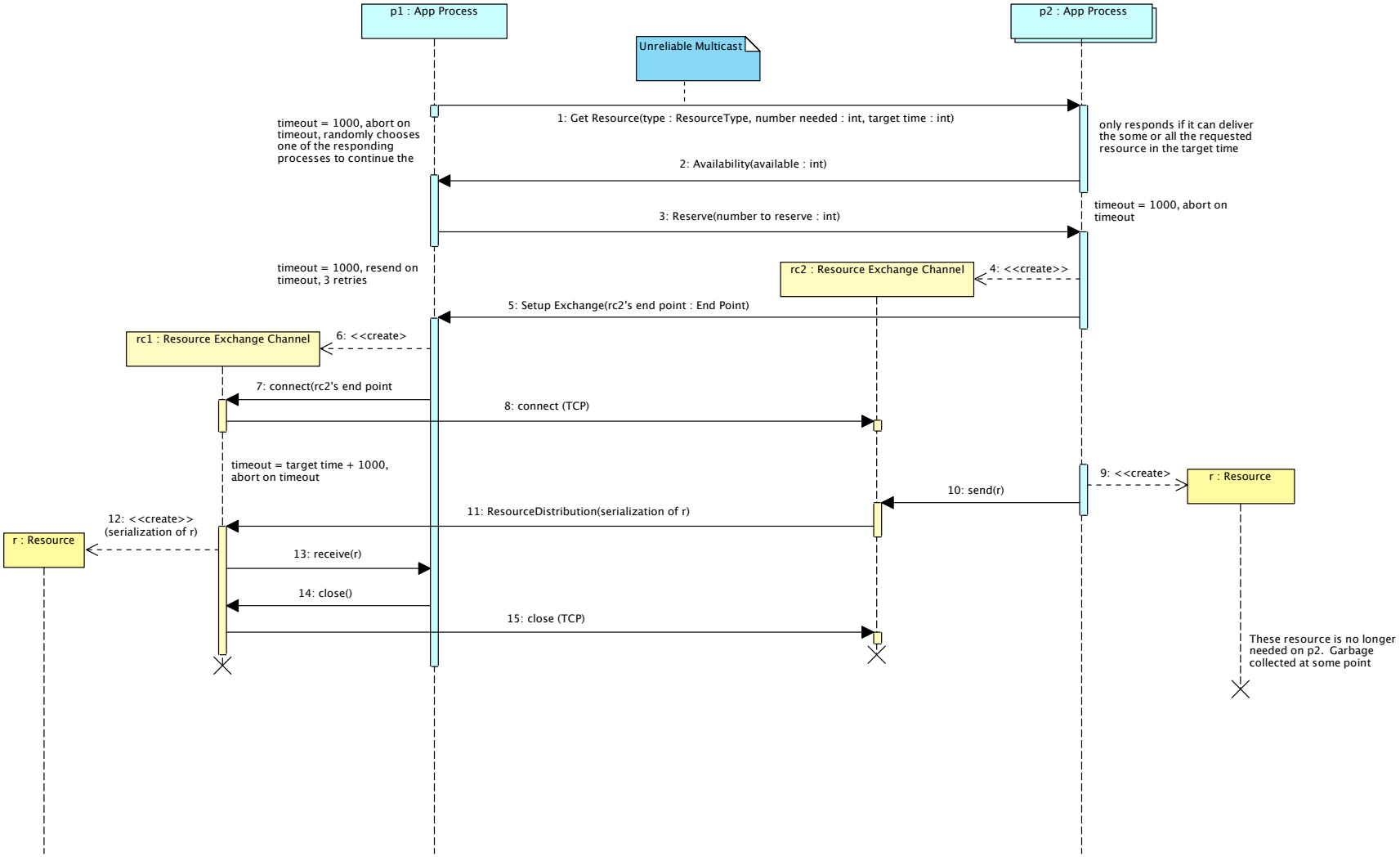

Thing-a-Bob Builder Factor
+Initialize()
Widget Builder Factor
+Initialize()
Gadget Assembler Conversation Factor
+Initialize()
Gadget Warehouse Conversation Factor
+Initialize()
Monitor Conversation Factor
+Initialize()
Registry Conversation Factor
+Initialize()
<<abstract>>
Conversation Factory
-typeMapping
+CommSubsystem
+Initialize()
+CreateFromMessage(Envelope) : Conversation
+CreateFromConversationType(Type) : Conversation
#Add(MessageType, ConvType)
<<create>>
<<abstract>>
Conversation
+CommSubsystem : CommSubsystem
+RemoteEndPoint : PublicEndPoint
+MaxRetries
+Timout
+PostExecutionAction : AcutionHandl...
+PreExecutionAction : ActionHandler
-lastSentEnv
-retries
-done : int
-error : Error
+ExecuteDetails()
+Launch()
#Initialize()
#ExecuteDetails()
#ReliableSend()
#IsEnvelopeValid(env, allowedTypes)
#UseConversationQueue() : bool
-ReliableSend()
<<abstract>>
Initiator Conversation
#Initialize()
#CreateFirstMessage()
<<abstract>>
Responder Conversation
+IncomingEnv : Envelope
+Initialize()
Initiator RR Conversation
#ExecuteDetails()
#ExpectedReplyType()
#ProcessValidReply()
Initiator Registration Conservation
#ExpectedReplyType()
#ProcessValidReply()
Initiator Subscribe Monitor Conversation
+ExpectedReplyType()
+ProcessValidReply()
Initiator Unsubscribe Monitor Conversation
#ExpectedReplyType()
#ProcessValidReply()
Initiator Resource Exchange
+ResourceType : ResourceType
+NumberWanted : int
+TargetMilliseconds : Int
+ResourcesReceived : List<Resource>
-dataChannel : Resource Exchange Channel
-numberToReceive : int
+ExecuteDetails()
Initiator Status Conversation
+ExecuteDetails()
Initiator Shutdown Conversation
+ExecuteDetails()
Responder RR Conversation
Responder Registration Conversation
+ExecuteDetails()
Responder Subscribe Monitor Conversation
+ExecuteDetails()
Responder Unsubscribe Monitor Conversation
+ExecuteDetails()
Responder Resource Exchange
-dataChannel : Resource Exchange Channel
+ExecuteDetails()
Responder Status Conversation
Responder Shutdown Conversation
+ExecuteDetails()

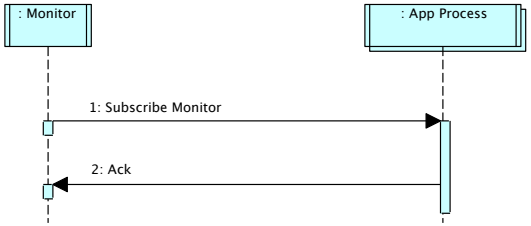

Control Message
-messageId : MesssageId
-convId : MessageId
+Decode(byte []) : Control Message
+Encode() : byte []
+InitMessageAndConversationIds()
+SetMessageAndConversationId(msgId : MessageId, convId : MessageId)
Stream Message
Register
-ProcessType : ProcessType
Process Info
+ProcessId : int
Subscribe Monitor
Unsubscribe Monitor
Status
-attribute
Shutdown
Ack
Nak
+Error : Error
Get Resource
+ResourceType : ResourceType
+NumberNeeded : int
+TargetMilliseconds : int
Availability
+Available : int
Reserve Resource
+NumberToResource : int
Setup Exchange
+TcpEndPoint : PublicEndPoint

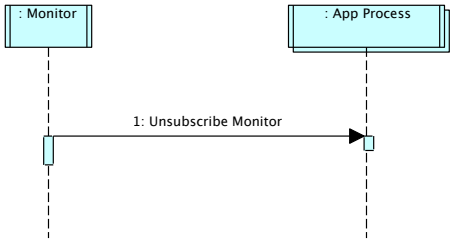

Resource Type
1
0..*
Resource
+SerialNumber : Guid
+BuildStartTime : DateTime
+BuildEndTime : DateTime
PublicEndPoint
+Host : string
+Port : int
+HostAndPort : string
+IpEndPoint : IPEndPoint
+LookupAddress(host : string) : IPAddress
+ToString()
Gadget
1
8
Widget
+Weight : int
Thing-a-Bob
+Height : int
-Width : int
1
15
MessageId
+Pid : int
+Seq : int
+Create()
+ToString()
+Clone()
+Compare()
<<Enum>>
ProcessType
Error
+Text : string
-Timestamp : DateTime
<<Singleton>>
Local Process Info
+ProcessId : int
+StartTime : DateTime

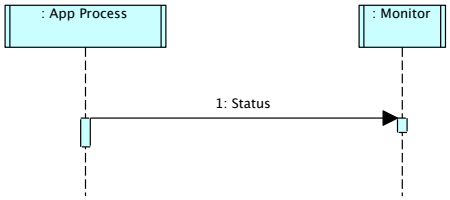

Responder Resource Exchange Conversation
creation with Incoming Env
Not Initialized
@Launch() /
setup conversation queue
Working
Deciding whether to respond
[Will not respond]
[Will respond, with n available] /
Create and send Availability Message
Save envelope in lastSentEnv
retries = MaxRetries
waiting for Reservation message
[retries = 0] On Timeout
[Received Reservation message,
but requested resources are not longer available] /
create and send back a Nak
[retries>0] On Timeout
[Received reservation message and
request resources still available] /
reserve the requested resources
retries = MaxRetries
Waiting for Setup message
[retries = 0] On Timeout
Failed
[retries>0] On Timeout
[Received Setup Exchange message]
Waiting for Exchange Channel Connection
On Timeout
@Dispose()
[reserved resources not available]
On connection
[reserved resources available]
On connection
Waiting for Resources to be made
[reserved resources availabe]
Sending Resources
[all resources sent]
Successful
@Dispose()

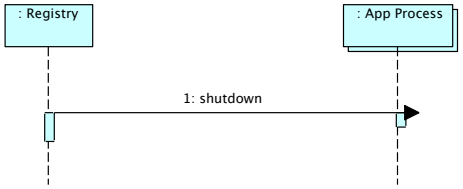

: App Process
: Registry
1: Register
2: Process Info

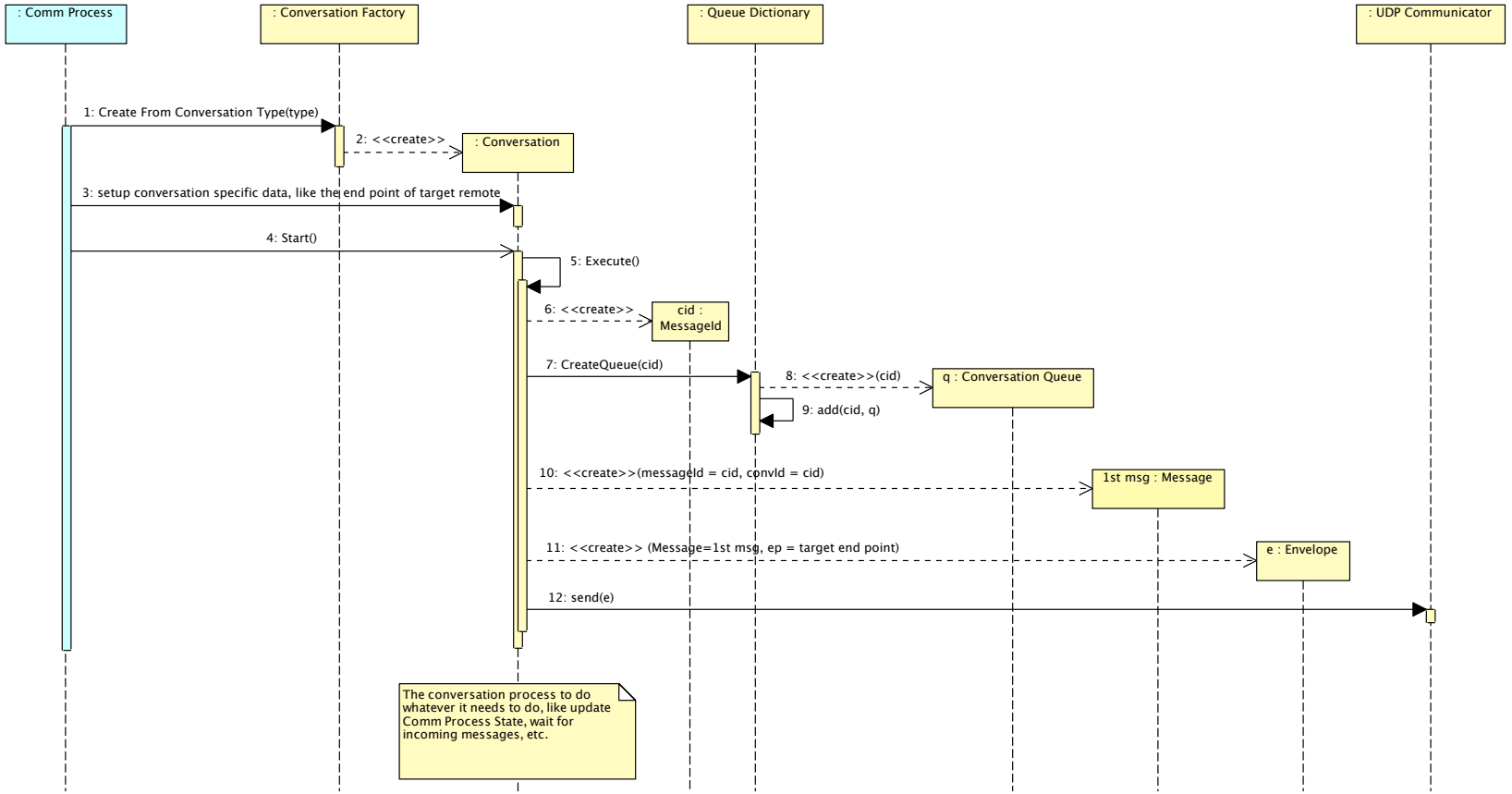

p1 : App Process
Unreliable Multicast
p2 : App Process
timeout = 1000, abort on timeout, randomly chooses one of the responding processes to continue the
1: Get Resource(type : ResourceType, number needed : int, target time : int)
only responds if it can deliver the some or all the requested resource in the target time
2: Availability(available : int)
timeout = 1000, abort on timeout
3: Reserve(number to reserve : int)
timeout = 1000, resend on timeout, 3 retries
rc2 : Resource Exchange Channel
4: <<create>>
5: Setup Exchange(rc2's end point : End Point)
rc1 : Resource Exchange Channel
6: <<create>
7: connect(rc2's end point
8: connect (TCP)
timeout = target time + 1000, abort on timeout
9: <<create>
r : Resource
10: send(r)
11: ResourceDistribution(serialization of r)
12: <<create>> (serialization of r)
r : Resource
13: receive(r)
14: close()
15: close (TCP)
These resource is no longer needed on p2. Garbage collected at some point

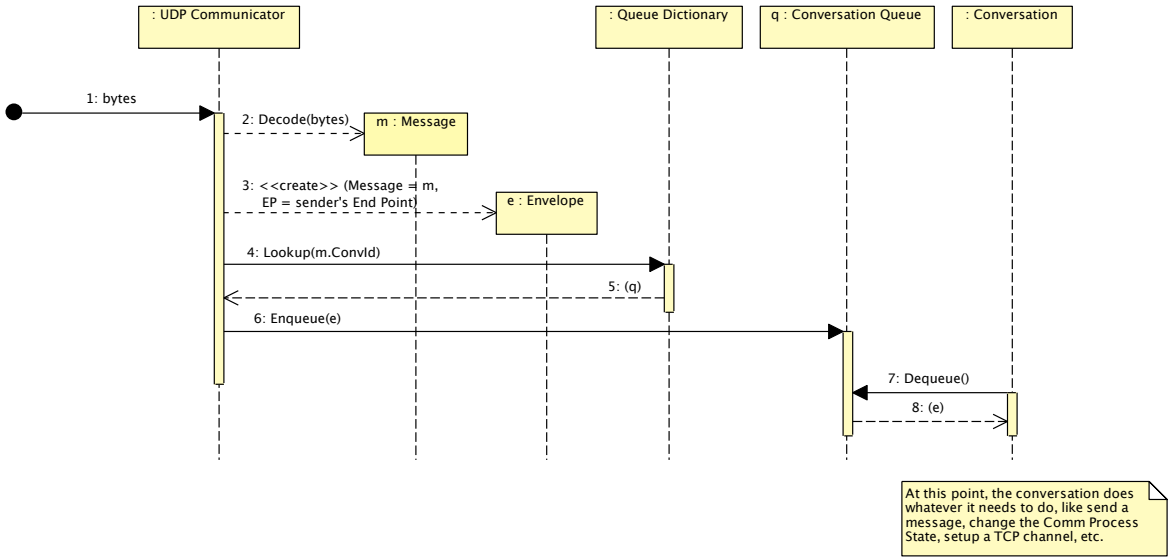

: Monitor
: App Process
1: Subscribe Monitor
2: Ack

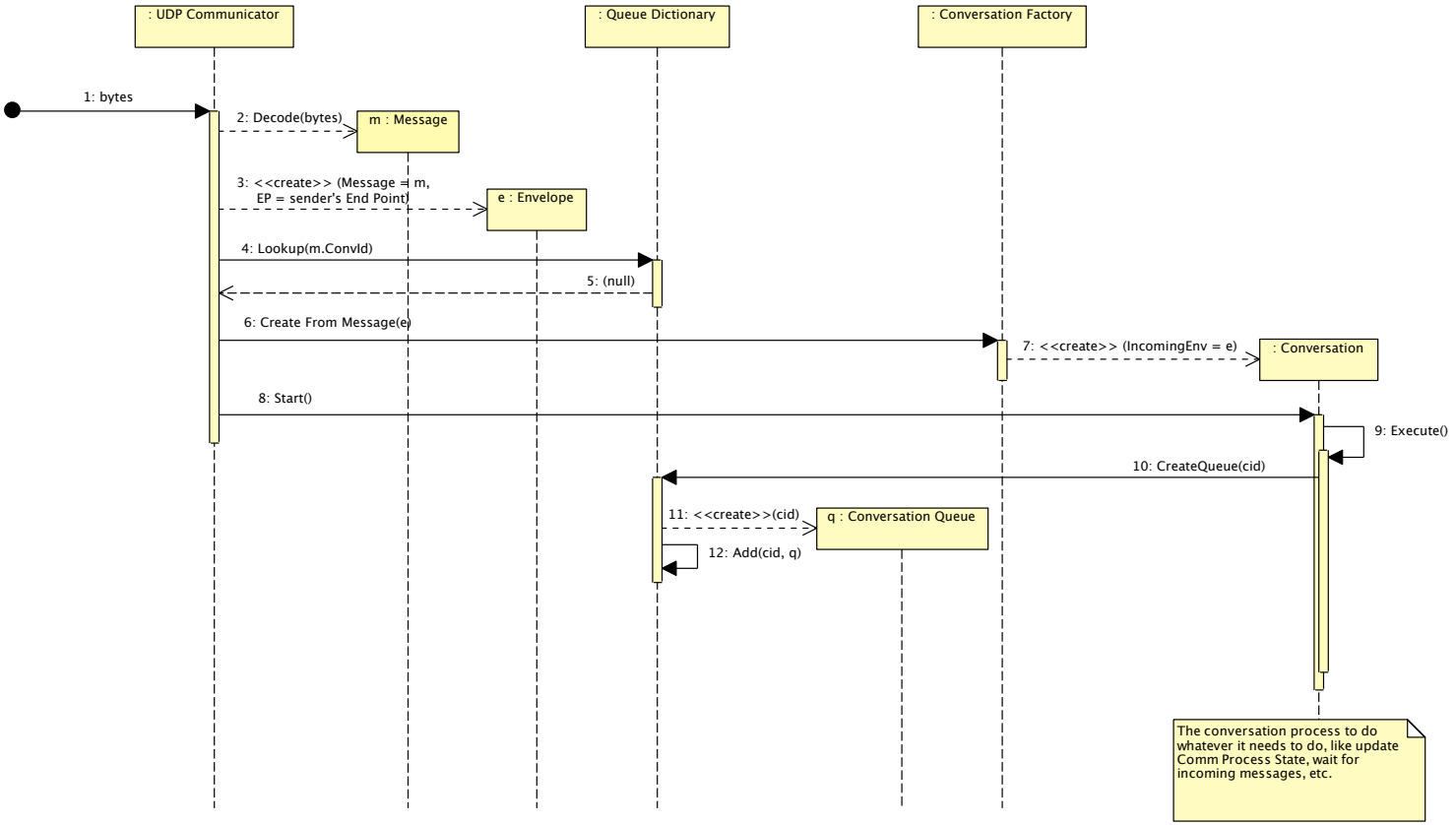

: Monitor
: App Process
1: Unsubscribe Monitor

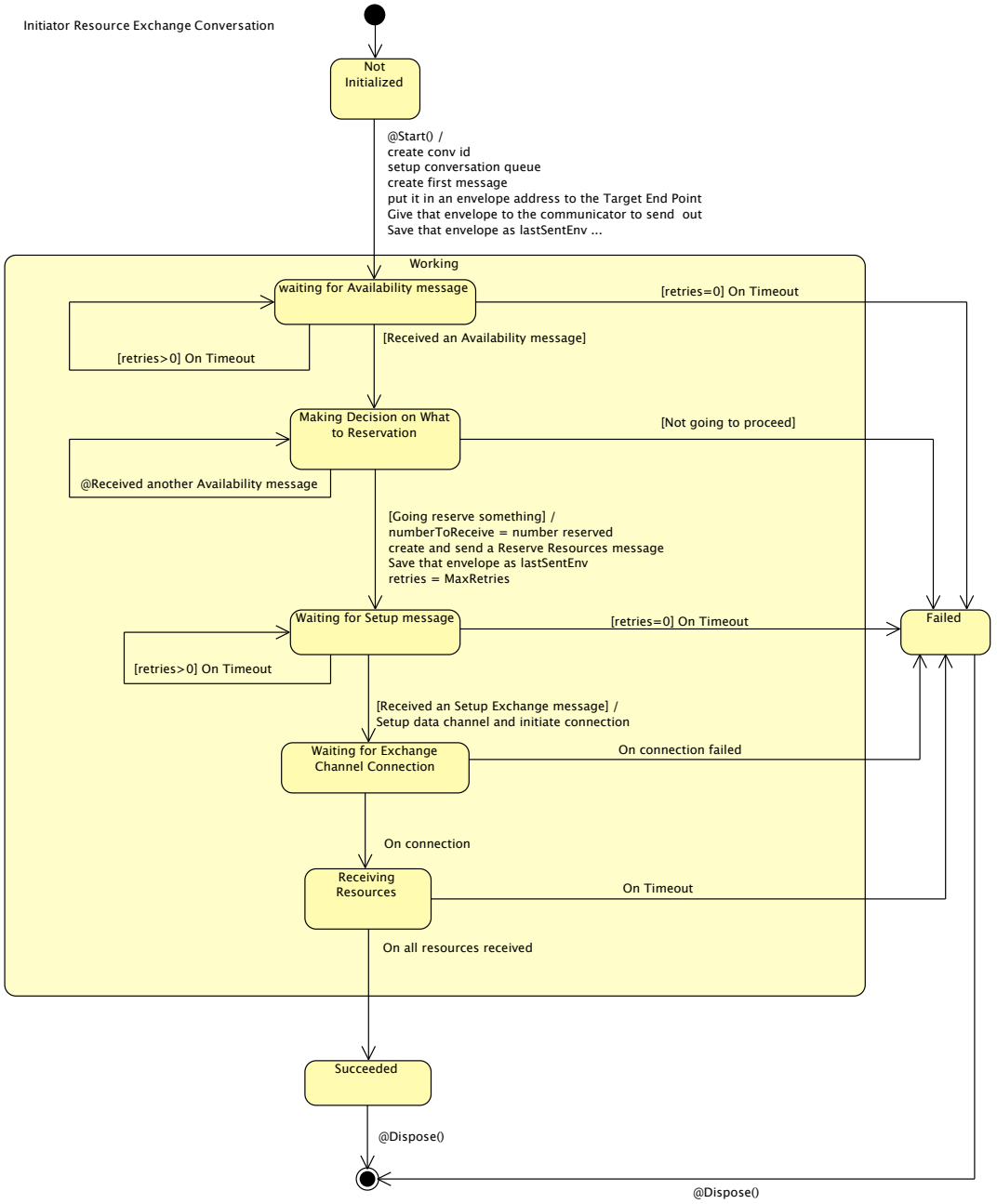

: App Process
: Monitor
1: Status


: Registry
: App Process
1: shutdown


: Comm Process
: Conversation Factory
: Queue Dictionary
: UDP Communicator
1: Create From Conversation Type(type)
2: <<create>>
: Conversation
3: setup conversation specific data, like the end point of target remote
4: Start()
5: Execute()
6: <<create>>
cid : MessageId
7: CreateQueue(cid)
8: <<create>>(cid)
q : Conversation Queue
9: add(cid, q)
10: <<create>>(messageId = cid, convId = cid)
1st msg : Message
11: <<create>> (Message=1st msg, ep = target end point)
e : Envelope
12: send(e)
The conversation process to do whatever it needs to do, like update Comm Process State, wait for incoming messages, etc.


: UDP Communicator
: Queue Dictionary
q : Conversation Queue
: Conversation
1: bytes
2: Decode(bytes)
m : Message
3: <<create>> (Message = m,
EP = sender's End Point)
e : Envelope
4: Lookup(m.ConvId)
5: (q)
6: Enqueue(e)
7: Dequeue()
8: (e)
At this point, the conversation does whatever it needs to do, like send a message, change the Comm Process State, setup a TCP channel, etc.


: UDP Communicator
: Queue Dictionary
: Conversation Factory
1: bytes
2: Decode(bytes)
m : Message
3: <<create>> (Message = m,
EP = sender's End Point)
e : Envelope
4: Lookup(m.ConvId)
5: (null)
6: Create From Message(e)
7: <<create>> (IncomingEnv = e)
: Conversation
8: Start()
9: Execute()
10: CreateQueue(cid)
11: <<create>>(cid)
q : Conversation Queue
12: Add(cid, q)
The conversation process to do whatever it needs to do, like update Comm Process State, wait for incoming messages, etc.


Initiator Resource Exchange Conversation
Not Initialized
@Start() /
create conv id
setup conversation queue
create first message
put it in an envelope address to the Target End Point
Give that envelope to the communicator to send out
Save that envelope as lastSentEnv ...
Working
waiting for Availability message
[retries=0] On Timeout
[retries>0] On Timeout
[Received an Availability message]
Making Decision on What to Reservation
[Not going to proceed]
@Received another Availability message
[Going reserve something] /
numberToReceive = number reserved
create and send a Reserve Resources message
Save that envelope as lastSentEnv
retries = MaxRetries
Waiting for Setup message
[retries=0] On Timeout
Failed
[retries>0] On Timeout
[Received an Setup Exchange message] /
Setup data channel and initiate connection
Waiting for Exchange Channel Connection
On connection failed
On connection
Receiving Resources
On Timeout
On all resources received
Succeeded
@Dispose()
@Dispose()