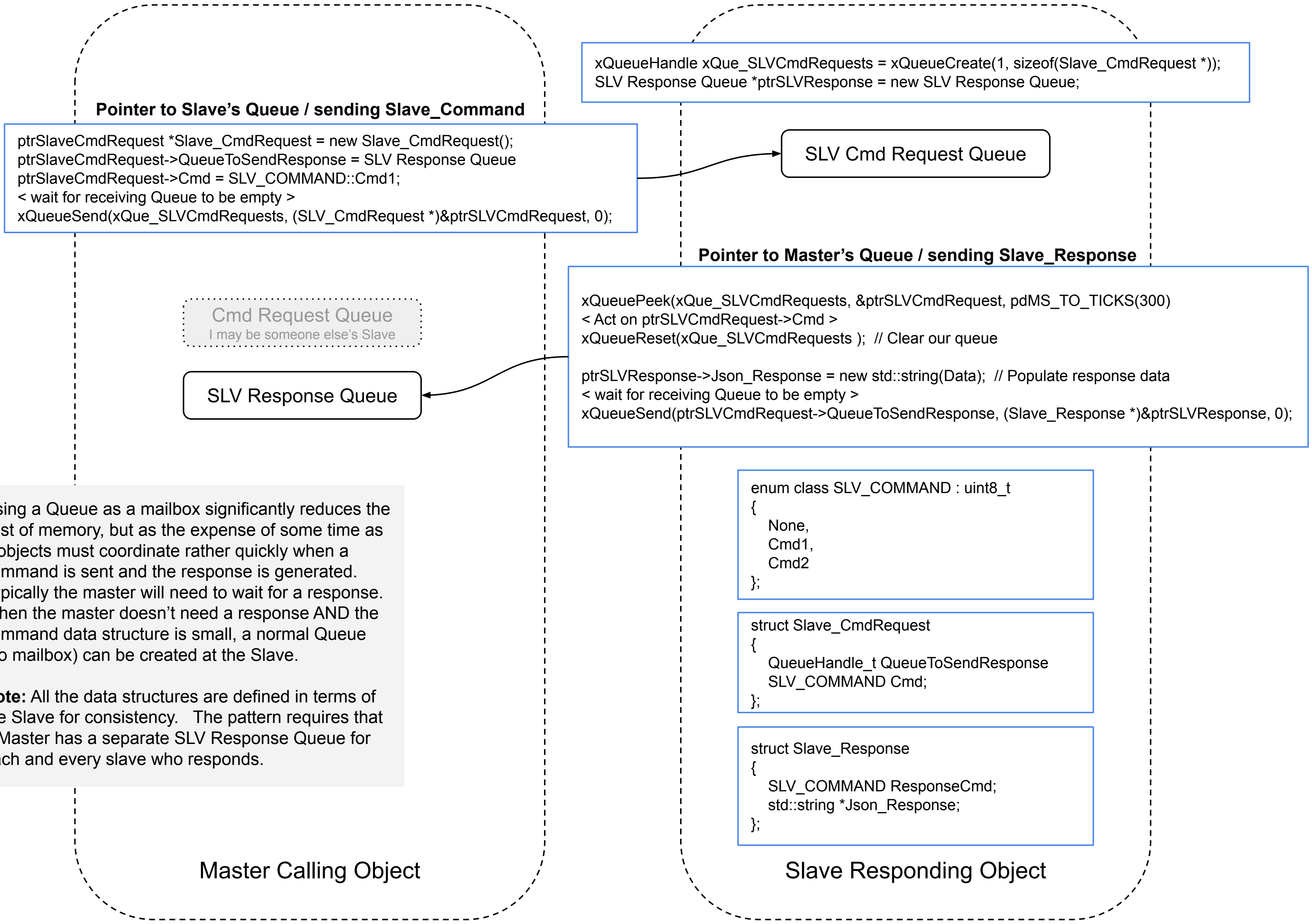
Queue-Mailbox Pattern

Master/Slave Relationship



```
xQueueHandle xQue_SLVCmdRequests = xQueueCreate(1, sizeof(Slave_CmdRequest *));
SLV Response Queue *ptrSLVResponse = new SLV Response Queue;
```

Pointer to Slave's Queue / sending Slave_Command

```
ptrSlaveCmdRequest *Slave_CmdRequest = new Slave_CmdRequest();
ptrSlaveCmdRequest->QueueToSendResponse = SLV Response Queue
ptrSlaveCmdRequest->Cmd = SLV_COMMAND::Cmd1;
< wait for receiving Queue to be empty >
xQueueSend(xQue_SLVCmdRequests, (SLV_CmdRequest *)&ptrSLVCmdRequest, 0);
```

Pointer to Master's Queue / sending Slave_Response

```
xQueuePeek(xQue_SLVCmdRequests, &ptrSLVCmdRequest, pdMS_TO_TICKS(300)
< Act on ptrSLVCmdRequest->Cmd >
xQueueReset(xQue_SLVCmdRequests );  // Clear our queue

ptrSLVResponse->Json_Response = new std::string(Data);  // Populate response data
< wait for receiving Queue to be empty >
xQueueSend(ptrSLVCmdRequest->QueueToSendResponse, (Slave_Response *)&ptrSLVResponse, 0);
```

```
enum class SLV_COMMAND : uint8_t
{
    None,
    Cmd1,
    Cmd2
};
```

```
struct Slave_CmdRequest
{
    QueueHandle_t QueueToSendResponse
    SLV_COMMAND Cmd;
};
```

```
struct Slave_Response
{
    SLV_COMMAND ResponseCmd;
    std::string *Json_Response;
};
```

Using a Queue as a mailbox significantly reduces the cost of memory, but as the expense of some time as 2 objects must coordinate rather quickly when a command is sent and the response is generated. Typically the master will need to wait for a response. When the master doesn't need a response AND the command data structure is small, a normal Queue (no mailbox) can be created at the Slave.

Note: All the data structures are defined in terms of the Slave for consistency. The pattern requires that a Master has a separate SLV Response Queue for each and every slave who responds.