Need for easy and effective XA support

Contents

- Transactional Messaging
- X/Open DTP Model
- Solutions for AMQP XA Support
- Dtx Class
- Observations
- Considerations

Transactional Messaging Semantic

Message Production

- A message is produced within the scope of a transaction
  - if transaction **commits** then the message is **enqueued**
  - if transaction **rolls-back** then the message is **discarded**

Message Consumption

- A message is consumed within the scope of a transaction
  - if transaction **commits** then the message is **discarded**
  - if transaction **rolls-back** then the message is **re-enqueued**

Messaging Most Common Use Case

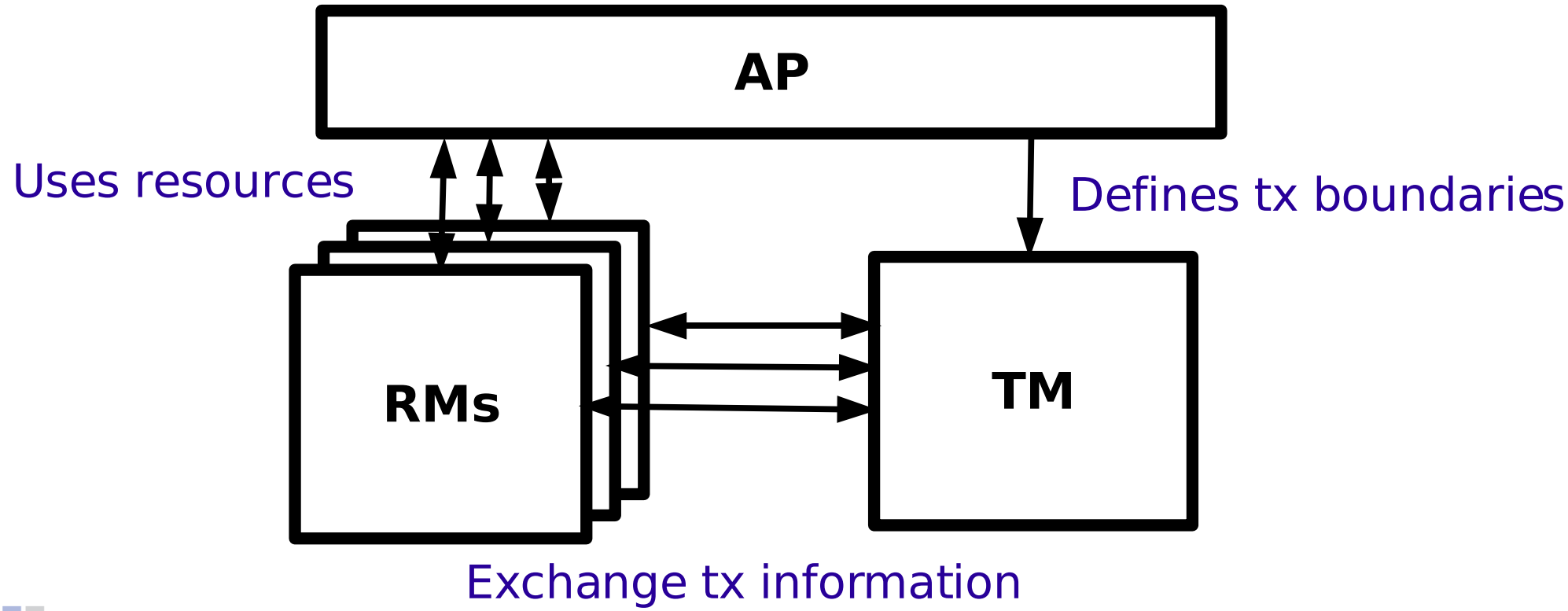
1. start distributed transaction T1
2. consume a message from Queue X
3. write to relational database
4. produce a new message on Queue Y (which may or may not be on the same broker process)

T1

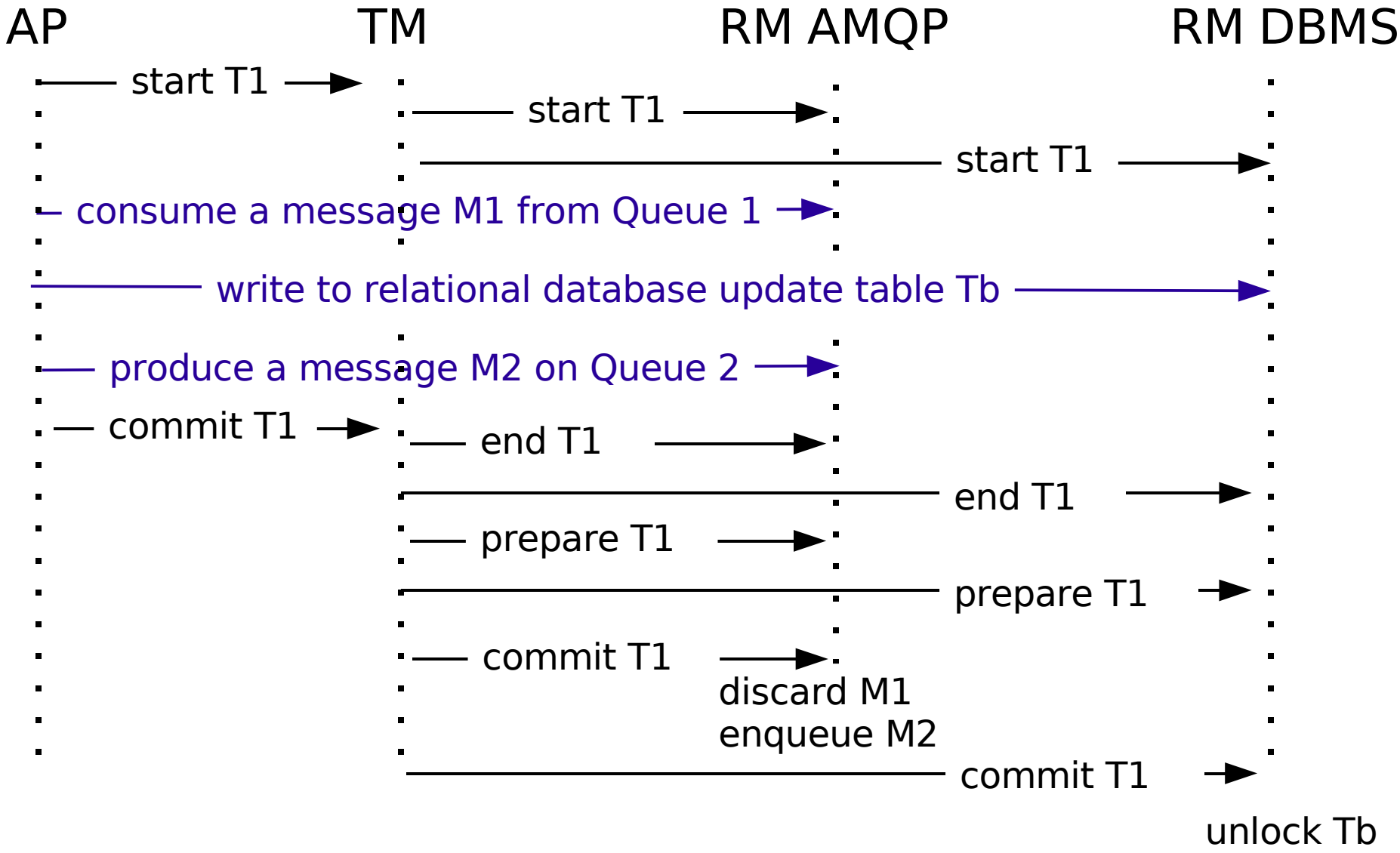
5. commit or rollback T1

X/Open Distributed Transaction Processing (DTP) Model

- Application Program (**AP**)
- Resource Managers (**RM**)
- Transaction Manger (**TM**)



X/Open DTP Model Messaging Most Common Use Case



Solutions for AMQP XA Support

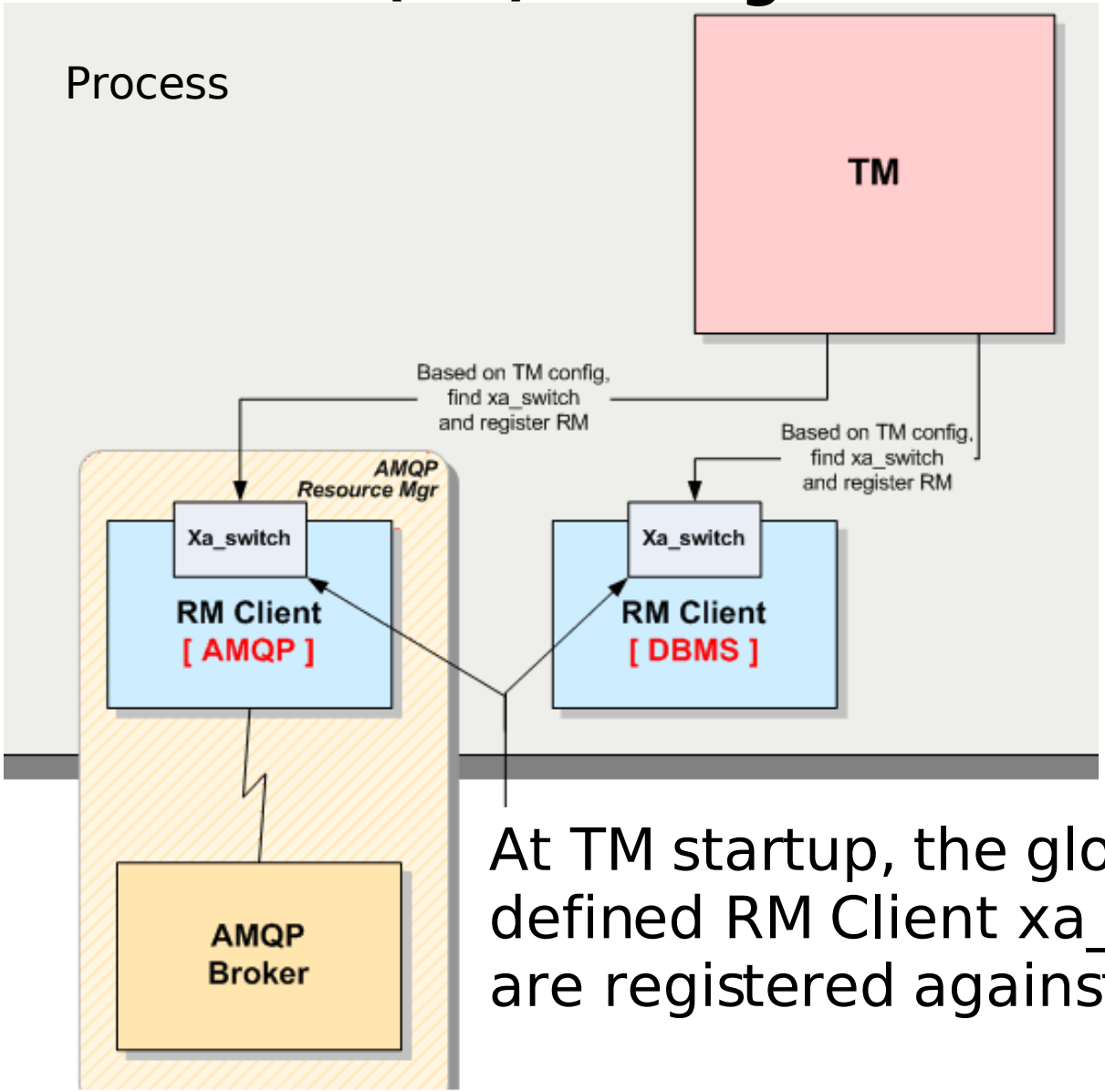
- Use some object communication facilities (RMI, CORBA, ..)
  - Additional transport layer that may not be as flexible and light way than AMQP
  - Additional configuration would be required
  - Interoperability may be broken as AMQP XA communication protocol would be left as an implementation choice
- Full control of distributed transactions over AMQP
  - Improve interoperability
  - Aid implementations that wish to provide XA support

Proposed Solution

- Extend AMQP dtx class to provide support for the X-Open XA architecture

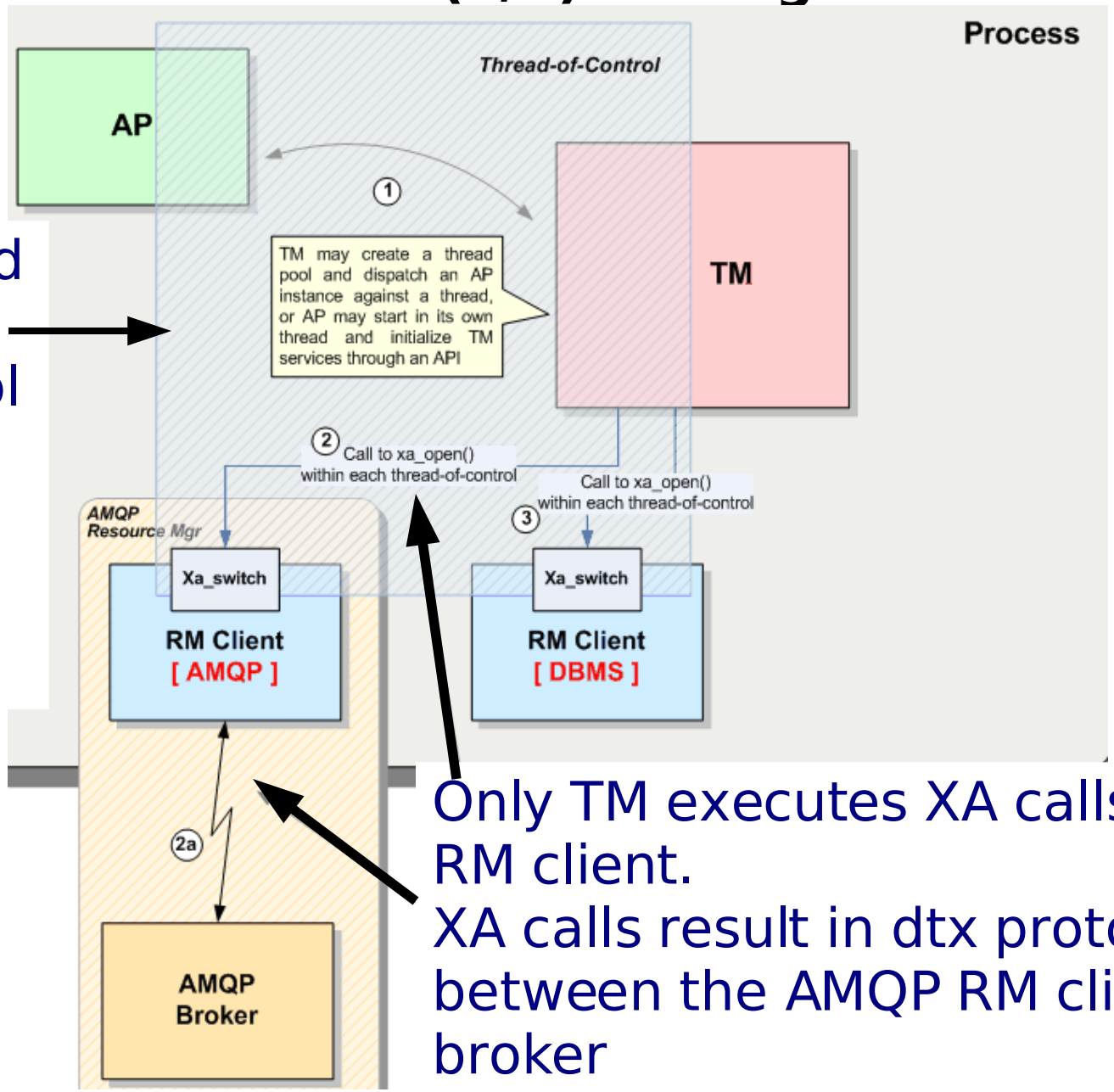
AMQP broker that wants to participate in global transaction has to be XA compliance Resource

C++ Use Case (1/4) RM registration with TM



At TM startup, the globally defined RM Client xa_switches are registered against the TM

C++ Use Case (2/4) RM registration with TM



Thread of control

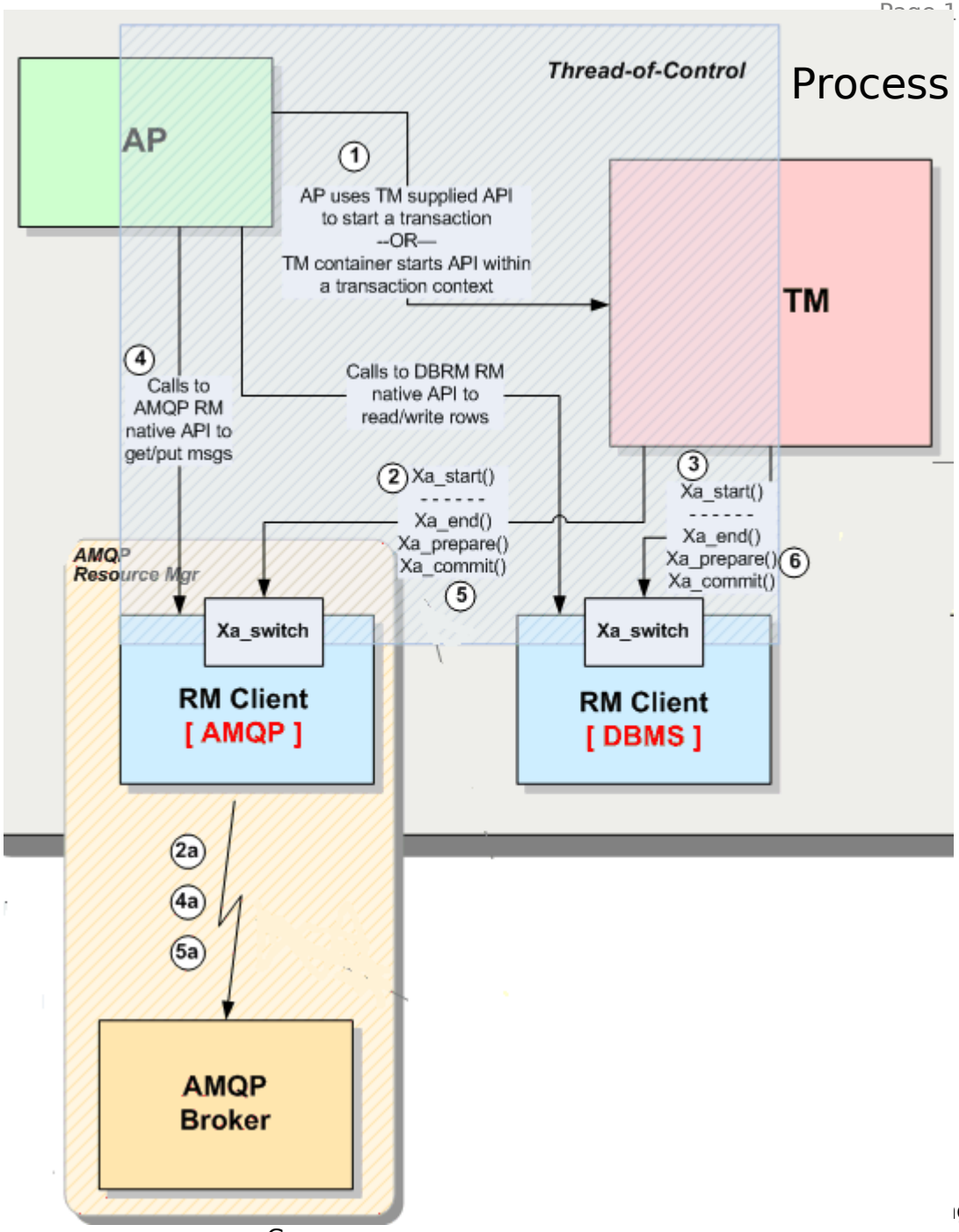
Only TM executes XA calls against an RM client.
XA calls result in dtx protocol flow between the AMQP RM client and broker

C++ Use Case (3/4) RM registration with TM

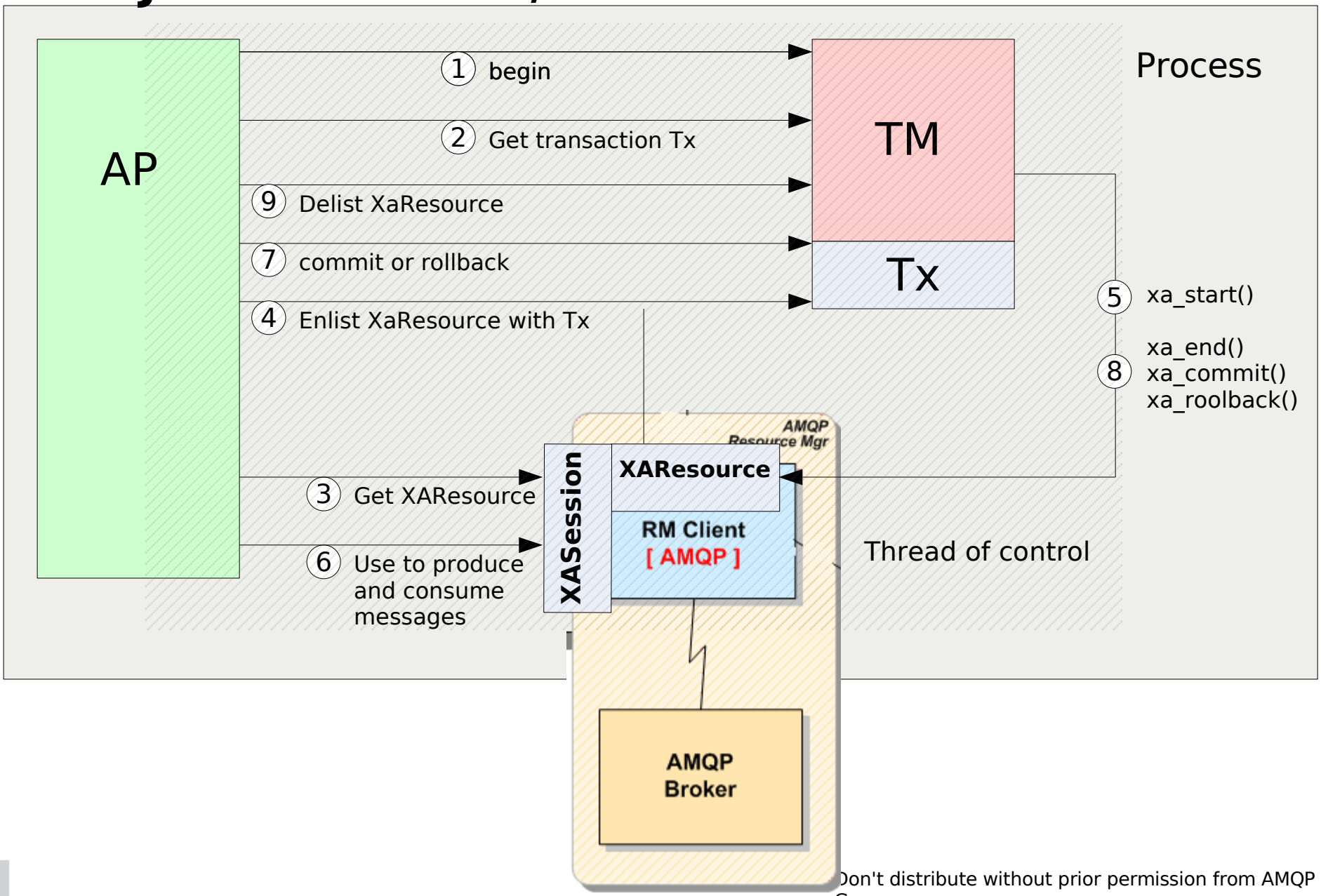
- The RM client is required to manage a mapping between channel and thread-context
  - AP, TM and RM client execute under the same thread of control
  - TM always calls xa_open If supported by RM client
  - TM always passes rmid with xa operations. rmid uniquely identifies the called RM instance with the Thread of control
- With this mapping RM client is able to determine which AMQP channel to employ based on the thread context
- It is likely that a RM client would maintain a pool of channel

C++ Use Case (4/4)

- The AP uses the native AMQP interface of the RM AMQP client for producing/consuming messages
- The RM AMQP client native interface is not defined by the AMQP Specifications
- Based on the thread context the RM AMQP client Map calls to the corresponding channel



Java Use Case 1/2



Java use case 2/2

- XAResource and XASession objects share the same AMQP channel
- There is a one to one association of a channel and a thread context (A JMS session is single threaded)
- Remark: several XAResources can be registered with the same transaction

dtx Class New Methods

- **select:** sets the channel to use distributed transactions
- **start:** messages are produced and consumed on behalf a transaction branch
- **Suspend:** suspend the currently running transaction branch
- **prepare:** prepares for commitment any message produced or consumed on behalf a transaction branch (default currently running)
- **commit:** commits the work associated with a transaction branch (default currently running)
- **rollback:** rolls back the work associated with a transaction branch (default currently running)
- **setTimeout:** sets the transaction timeout in seconds
- **getTimeout:** get the current transaction timeout in seconds
- **recover:** obtains a list of transaction branches that are in a prepared or heuristically completed state.
- **forget:** forgets about a heuristically completed transaction branch

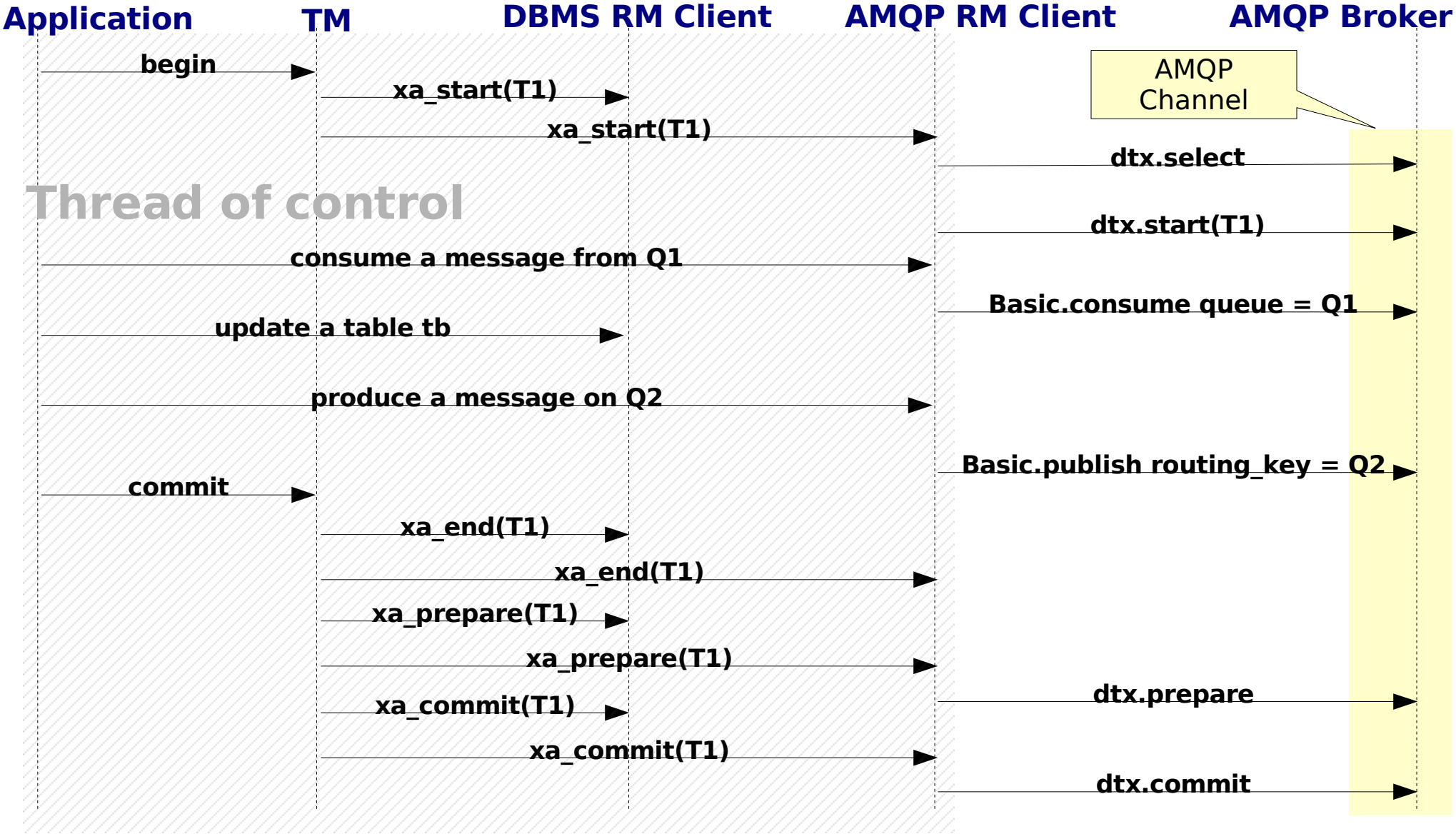
dtx-Defined Domain

XID contains a format identifier, two length fields and a data field:

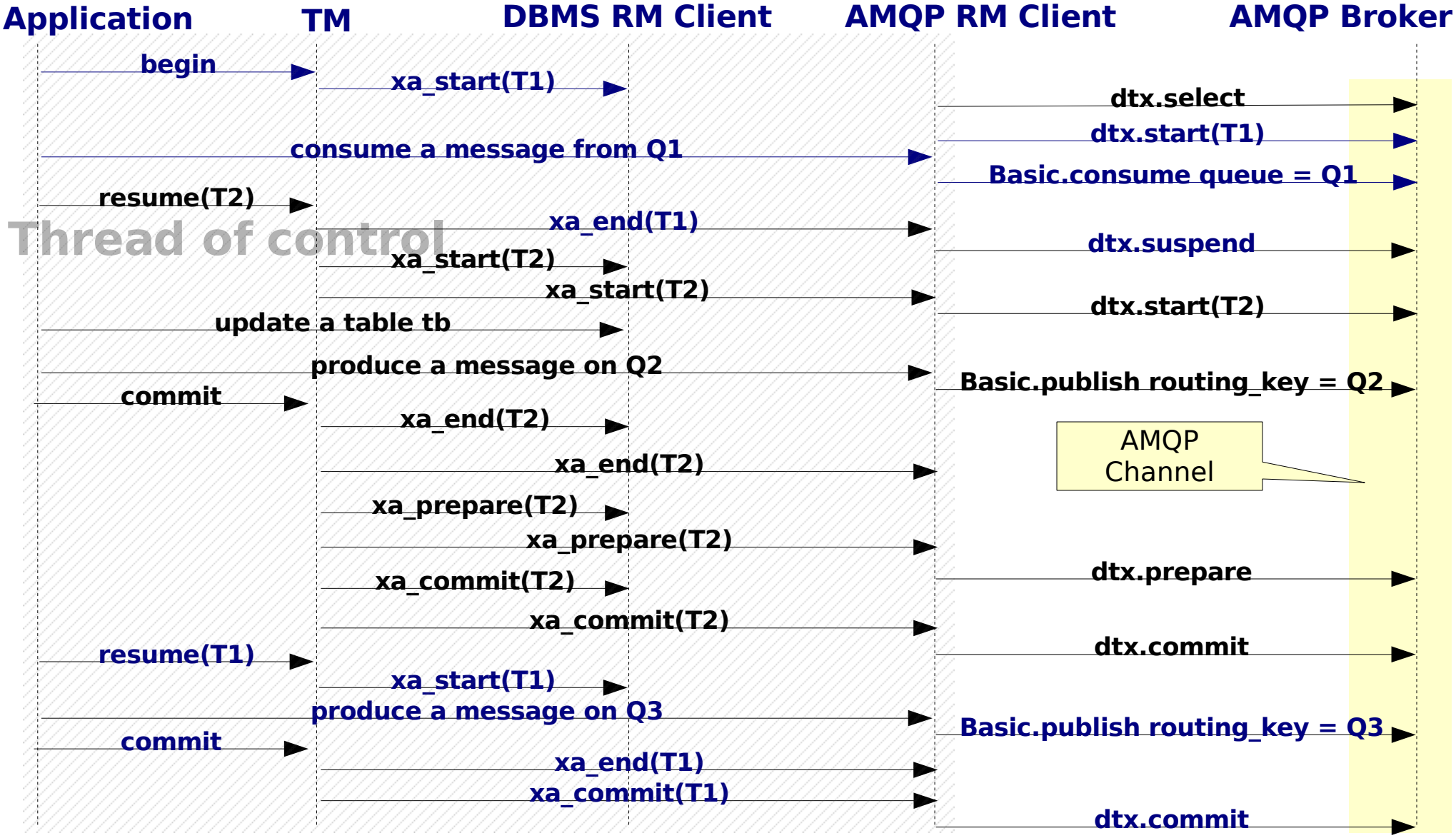
- long formatID
- octet gtrid_length
- octet bqual_length
- table data[128] : a table of 128 bytes

Note that XIDs can be null

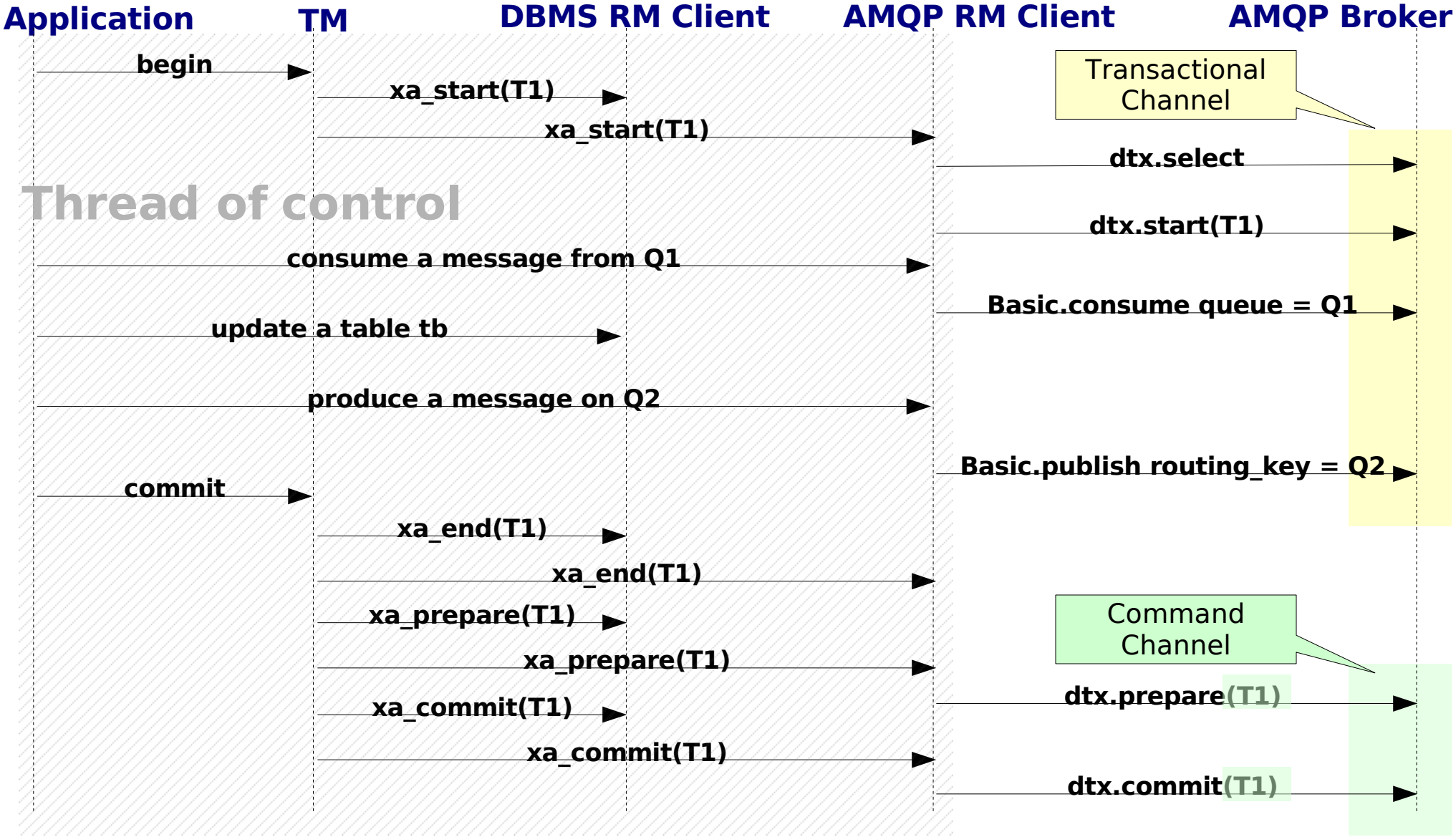
Messaging Most Common Use Case



Messaging Extended Use Case



Using a Command Channel



dtx Class Grammar

```
dtx             = C:select S:select-ok *xaWork
                | *xaMonitor

xaMonitor       = xaOutcome
                |  xaRecovery
                | C:setTimeout S:resp
                | C:getTimeout S:resp

xaWork          = C:start S:resp *xaMoreOps xaOutcome
                | xaMonitor

xaMoreOps       = C:suspend S:resp xaWork

xaOutcome       = xaRollback
                 | xa1PhaseCom
                 | xa2PhaseCom

xaRollback      = C:rollback S:resp

xa1PhaseCom     = C:commit(TMONEPHASE) S:resp

xa2PhaseCom     = C:prepare S:resp C:commit S:resp
                 | C:prepare S:resp C:rollback S:resp

xaRecovery      = C:recover S:recover-resp
                | xaOutcome
                | C:forget S:resp
```

Observations 1/2

- There is a one-to-one association between thread-context and channel
- More than one channel CAN be used
  - Transactional channel
    - Any works performed between dtx.start/dtx.suspend or dtx.start/dtx.commit is done on behalf a transaction branch identified by XID.
  - Command Channel
    - can be used for any XA operations other than start and suspend under the condition that the transaction XID is specified
- If the transaction context is not specified then the currently running transaction branch is selected

Observations 2/2

- All the dtx operations have an access-ticket to prevent unauthorized use
- The purpose of the dtx.select operation is for the server to optimize handling of distributed transaction. Once a channel is selected it cannot be disassociated with XA support
- As we expect that the sever will setup some mechanisms for handling distributed transactions that will result in some kind of overhead we do not recommend using an XA channel for non-transacted traffic
- The dtx class is optional