Connection switch improvements

Problems

- State of switch is unknown to chat, making it unclear if switch is progressing or stuck.
- Multiple connection switches can be triggered before the first one completes, leaving unused queues.

Solution

Propagate switch state to UI via extended `ConnectionStats`. UI will display switch status if switch is in progress, the button to initiate switch will be disabled depending on switch state.

Agent backend

Add following fields to `connections` table:

```
sql ALTER TABLE connections ADD COLUMN rcv_switch_status TEXT;
ALTER TABLE connections ADD COLUMN snd_switch_status TEXT;
```

We can use either existing `SwitchPhase` type as status, or new more detailed types separate for rcv and snd switch.

```haskell data RcvSwitchStatus = RSSQueuingSwch -- set in beginning of switchConnectionAsync' before queueing SWCH command | RSSSwchStarted -- set in beginning of switchConnection' | RSSQueuingQADD -- set in switchConnection' before queuing QADD | RSSSentQADD -- set in runSmpQueueMsgDelivery after receiving Right in response to sending QADD | RSSReceivedQKEY -- set on receiving QKEY, in beginning of qKeyMsg | RSSQueueingSecure -- set before queueing ICQSecure in qKeyMsg | RSSSecureStarted -- set in beginning of ICQSecure processing in runCommandProcessing | RSSQueueingQUSE -- set in ICQSecure processing before queueing QUSE | RSSMessageReceived -- set after receiving first message in the new queue (processSMPTransmission, setRcvQueuePrimary) | RSSQueueingDelete -- set before queuing ICQDelete in processSMPTransmission | RSSDeleteStarted -- set in beginning of ICQDelete processing in runCommandProcessing

-- after ICQDelete processing rcvswitchstatus is set back to NULL, on internal errors as well

canStopRcvSwitch :: RcvSwitchStatus -> Bool canStopRcvSwitch = \case RSSSentQADD -> True _ -> False ```
```

I don't know use for most of the statuses yet, except for debugging, and more granular control over converting switch state into UI representation.

At least it seems necessary to record the very beginning of switch (async command, sync command), points of sending QADD and receiving QKEY, and to reset after ICQDelete. `SwitchPhase` type seems to be insufficient.

When it is in progress, repeatedly switching connection in UI is only allowed in RSSSentQADD status - to avoid race conditions with internal commands processing, and sender processing.

Logic of repeat switch of connection could be split into stopSwitchConnection and existing start switch command on client level (basically chat automating two buttons for user).

```
haskell stopSwitchConnection :: AgentErrorMonad m => AgentClient -> ConnId -> m ConnectionStats stopSwitchConnection c = do -- in transaction: -- if rcv_switch_status is RSSSentQADD: -- - deleteConnRcvQueue -- - reset rcv_switch_status to NULL -- else: throw error -- deleteQueue on server
```

Repeat switch would send repeat QADD to sender, sender should delete snd queue that was previously set to replace existing queue (check in qAddMsg).

Snd switch statuses:

```haskell data SndSwitchStatus = SSSReceivedQADD -- set on receiving QADD, in beginning of qAddMsg | SSSQueueingQKEY -- set in qAddMsg before queuing QKEY | SSSReceivedQUSE -- set on receiving QUSE, in beginning of qUseMsg | SSSQueueingQTEST -- set in qUseMsg before queuing QTEST | SSSSentQTEST -- set in runSmpQueueMsgDelivery after receiving Right in response to sending QTEST

-- after processing AMQTEST in runSmpQueueMsgDelivery, sndswitchstatus is set back to NULL ```

In case of send it is enough to know that "snd switch is in progress", as sender cannot do any actions with recipient's switch, so status granularity is not necessary, but it can be useful for debugging and UI representation.

---

Problem: If permanent errors in the new queue prevent switch from completing, it will be stuck forever and repeat switch will not be allowed. How to abort switch in this case while switch is in progress?

Solution:

- In case of a permanent error on recipient side, delete queue (deleteQueue on server, catching errors; deleteConnRcvQueue in database), reset rcvswitchstatus to NULL. However, if rcvswitchstatus is one of RSSMessageReceived, RSSQueueingDelete, RSSDeleteStarted, it means that sender has already deleted the original queue and will not be able to revert, so the recipient has to complete switch regardless of permanent errors.
- In case of a permanent error on sender side (e.g. AUTH error when trying to send QTEST), recipient has no way of knowing switch will
```

never complete - one option is for sender to send a new message "QERR SMPQueueInfo" (/ QFAIL) in the original queue and delete queue from database; after receiving QERR recipient tries to delete queue on server, deletes queue in database, resets rcvswitchstatus to NULL.

A new SPFailed SwitchPhase might be required for notifying client, so that switch failure is visible in client as a new chat item.

In case this new QERR message would have to be added anyway - why allow stopping/re-triggering switch at all? Sender agent could send QERR on permanent error when trying to send QKEY as well. This would allow to reduce number or required statuses, and remove the need to introduce stopSwitchConnection api / stop logic in agent.

Type to communicate switch status to chat and UI:

```haskell -- rename: -- ConnectionStats -> ConnectionInfo, -- getConnectionServers -> getConnectionInfo

data ConnectionInfo = ConnectionInfo { rcvServers :: [SMPServer], sndServers :: [SMPServer], rcvSwitchStatus :: Maybe RcvSwitchStatus, sndSwitchStatus :: Maybe SndSwitchStatus } ```

Existing connections:

- Should switch be allowed for existing connections with switch in progress?
- If switch should not be allowed, how to distinguish whether switch is in progress - by connection having multiple queues?
- If yes, how to account for connections with extra queues from unfinished switches?

It's the easiest to allow switch for existing connections with switch already in progress and/or multiple switching queues.It is currently allowed, so it's not breaking anything new.
```