TX²

Technical Solution:
Event Coordination and Control

Scheduling and Timing

- Initiate a flow at a specified time
- Wait for a specified number of seconds
    - Configurable delay
    - Thread safe / non-blocking to flow or other flows

See accompanying article on Timer Management with WBI-MB

Coordinating Multiple Events

The top level requirement is to provide functions / services within Message Broker to provide the following interfaces:

1. Provide a control group name and a list of ids in that group.
2. "Tick off" an item (within a specified group) when it has been processed.
3. Determine that all items within a specified group have been "ticked off" within a specified time.
4. If an "unexpected" item arrives that is not in a group we are tracking, then wait a specified period of time for the control group information; if it has not arrived in that time period, then error.

Restrictions:

- Must have a control list of what items are expected (but that list may arrive after the items themselves)
- Need unique item_ids (or at least to be able to construct unique ids from items)
- Need unique group_ids for each group.

(c) TX Squared Ltd. 2003-2006

Note: Users MAY need to store the original item data until the control set is complete (all items have arrived). To simplify processing, we have implemented two separate functions to enable store and retrieve of data (to avoid complicating the coordination and timer functions). This is simply a table (key, data, expire_time) and two sample functions : store and retrieve_and_delete …… The key to using this is that the Control list of items is the "success data" sent back by the timer when all the items have arrived; this can drive a flow that retrieves all the items by using the retrieve function and the list of items.

(c) TX Squared Ltd. 2003-2006

Outline Solution

- Create new **"Group Control" table**, use schema from aggregation table in broker database – create the table in the EAI database.
- **Define 2 compute nodes** for "Group Control" and "Item Tick-off". Users must set environment variables before these nodes (as they are NOT allowed to modify the compute nodes).
- Set environment to pass parameters to the compute nodes …… individual mark is just one record – the control parameters are an array of records – we have defined some environment names "Environment." ….. (including queue names for success / failure) – Variables:
    - "Group Control" compute node inputs (Environment.ControlNode.xxxxxx) :
        - Group_Control_ID
        - Item_ID [*]
        - Timeout_Timestamp
        - Select_Existing_Errors (select that aborts timer because of error – without waiting for the timeout)
        - Success_Trigger_Queue
        - Success_Data
        - Failure_Trigger_Queue
        - Failure_Data
    - "Item" Compute node inputs (Environment.ItemNode.xxxxxx) :
        - Item_ID
        - Timeout_Timestamp
        - Failure_Trigger_Queue
        - Failure_Data
- We have created a **TIMER database** table (in the EAI database) with the following columns:
    - End_Time
    - Success_Select
    - Success_Select_Count
    - Failure_Check_Select
    - Data_for_Success
    - Queue_for_Sucess
    - Data_for _Fail
    - Queue_for_Fail
- We have created a **java input node "Heartbeat"** – an input node that triggers every X seconds

(c) TX Squared Ltd. 2003-2006

- Then we have a **Timer flow** – responds to heartbeat and loops for each entry in TIMER database table:
  - Check if select statement 1 (if not NULL) returns a count of <Success_Select_Count> → good response (if not null)
  - Check if select statement 2 (if not NULL) returns a count > 0 → Failure response
  - Check if current time > timeout time → Failure response
  - (any of the above – delete the row from database)
  - Otherwise – leave in database … check next entry ….. when all checked, end flow.

- Logic for "Control Group" compute node: (Inputs C=Control Group, <existing_errors_select_statement>….)
  - Loop for each item in the array of item ids (I):
    - Select from CONTROL database where item_id = I
    - If found – delete the row from the table
    - If not found – insert row into table, Unsolicited = "N", Item_ID = I, Group_ID = C
  - End Loop
  - Insert row into timer database:
    - Timeout, "SELECT * from CONTROL where control_id = "C , <existing_errors_select_statement>, success Q, success data, fail Q, fail data
  - Exit

- Logic for "Item" compute node: (Inputs Item = X)
  - Select * from CONTROL where item_id = X
  - If found – delete row
  - If not found – Insert Row : Unsolicited = "Y", Control_ID = <NULL>, item_id = X
  - Set a timer (insert into timer database):
    - Timeout, "SELECT * from CONTROL where item_id = X", NULL, NULL, Failure Q, Failure Data

(c) TX Squared Ltd. 2003-2006