**Communication between Metapop (multiple instances), Metapop manager, and an R data manipulation engine:**

MPM = metapop manager

MMM = metamodel manager

RRR = JSON-RPC server implemented in R

MP = Ramas Metapop

1. User executes MetapopManager.exe. In the GUI for MPM, user specifies (1) the number of MP files to be run, (2) the names of the MP file(s) to be run as part of the metamodel, and (3) the name of the external processor (.EXE) containing the user-specified metamodel linkage functions (for now we are using a JSON-RPC server developed in R and set to initiate automatically upon calling R.EXE by first setting a local environment variable : R\_PROFILE for initiating the server). Finally, the user must specify (for each instance of MP) the list of global and population state variables that are needed for this metamodel. To simplify GUI development, this information (varList) may be supplied in the form of a text file?.
2. MPM starts RRR or other external processor/server
3. MPM spawns each specified instance of MP in RPC mode. In RPC mode, MP knows how to receive and send JSON\_RPC objects at the appropriate times. MPM assigns a unique Client ID to each spawned instance of MP.
4. At this point, each instance of MP essentially speaks directly with RRR (single instance of RRR) via stdin/stdout. MP makes the following calls to RRR, in this order:
   1. MP (in RPC mode) sends a command to RRR: "Initialize"
      1. MP sends Client ID#
      2. RRR does: initialize global variables to store the number of MP instances, etc. Also, RRR communcates a confirmatory response to MP.
      3. MP receives: Boolean confirmation of message receipt
   2. MP (in RPC mode) sends a command to RRR: "StartSimulation"
      1. MP sends: Client ID#, MetapopStateVarsGlobal. MetapopStateVarsGlobal object contains information that is held constant across the entire simulation (e.g., nyears, nsimulations, ncatastrophes, nstages, etc.).
      2. RRR does: Stores global state variables in memory. Initializes Modifier object which will be used later to change MP parameters, using the parameters from MetapopStateVarsGlobal to size the arrays. Also, RRR communcates a confirmatory response to MP.
      3. MP receives: Boolean confirmation of message receipt.
   3. {IF time step 0} Initiate simulation: MP (in RPC mode) sends a command to RRR: "StopTimeStep"
      1. MP sends: Client ID#, MetapopStateVarsPop. MetapopStateVarsPop object contains information about the initial state of the population (that is, initial abundances, vital rates and such) for use in modifying population parameters for all MP instances in the next time step.
      2. RRR does: Store population state variables in memory.
      3. MP receives: Boolean confirmation of message receipt.

----LOOP THROUGH YEARS---

* 1. MP (in RPC mode) sends a command to RRR: "StartTimeStep"
     1. MP sends: Client ID#, nothing more. Just serves as signal to RRR to send a Modifier object.
     2. RRR does: Implement routines that modify the parameters in this MP instance based on the previous-year state of the metamodel (state variables from all MP instances from the previous time step). NOTE: this is the step that involves user-specified linkages among the various instances of Ramas Metapop.
     3. MP receives: Modifier object (and Boolean confirmation of message receipt). The Modifier object contains parameter values that have been altered to reflect the state of the metamodel in the previous time step.
  2. MP (in RPC mode) sends a command to RRR: "StopTimeStep"
     1. MP sends: Client ID#, MetapopStateVarsPop. MetapopStateVarsPop object contains information about the current state of the population for use in modifying population parameters for all MP instances in the next time step.
     2. RRR does: Store population state variables in memory.
     3. MP receives: Boolean confirmation of message receipt.
  3. Repeat steps c and d asdirected by MP (number of years)
  4. Once all replicates have been finished, MP (in RPC mode) sends a command to RRR: "StopSimulation"
     1. MP sends: Client ID#
     2. RRR does: nothing
     3. MP receives: Boolean confirmation of message receipt.
  5. Finally, MP (in RPC mode) sends a "Finalize" command to MPM to indicate that the simulation has completed.