How to differentiate between a query and a process?

Process: Any method (of any module) which can be called by the other modules, and has been made a standard for the Brihaspati-4 LMS by the module developer, will be initially termed as a process.

Query: Any process whose response can be obtained immediately every time when an API call is done can be declared by the module developer as a query.

All the queries can also be called as a process. But, the reverse of this should never be done. All the processes must only be called by the process only. No process should be called as a query by any module.

Modules will only send one query/process at a time.

Acronyms of the different modules of Brihaspati-4:

// Glue Code = gc
// Authentication Manager = am
// Communication Manager = cm
// Indexing Manager = im
// Routing Manager = rm
// Web Server Module = ws
// Web Module = web
// VoIP = voip
// DFS = dfs
// UFS = ufs
// Mail = mail
// Message = sms

1) Queries:

Queries have to be sent to the Glue Code in the form of **messages only** which will be responded immediately using the API call.

All the modules can form the queries and send them to the Glue Code one by one in the following Format:

que.{query_with_parameters_in_their_correct_sequence}.{acronym_of_the_module_which_sends_the_query}

Example: Communication Manager can send this following query to the Glue Code:

que.getNodeID().cm

The Glue Code will see that it's a query from the "**que.**" and then check that to which module the query belongs to (by using the ModuleCheck class) and then the Glue Code will use the API call to get the response of the query from the module the method belongs to, and then send the response of the query to the corresponding module which initiated the query (Glue Code will get to know about this from the "**.cm"** present at the end of the query or the process which has to be preserved even after forming the response).

2) Processes:

Some processes will have a reply, others will not. The process (if responded) will be responded later after some time through the corresponding input buffer of the Glue Code.

Messages Form:

Processes in the form of messages can be formed and sent to the Glue Code in the same way as the queries are being sent to the Glue Code. The only difference is that the processes will start with "**pro.**" instead of "**que.**".

All the modules can form the processes in the form of messages and send them to the Glue Code one by one in the following Format:

pro.{process_with_parameters_in_their_correct_sequence}.{acronym_of_the_module_which_sends_the_process}

Example: Communication Manager can send this following process to the Glue Code: **pro.findNextHop(String 6FB23548386CCDB, int 2).cm**

The Glue Code will see that it's a process from the "**pro.**" and then check that to which module the process belongs to (by using the ModuleCheck class), and then the Glue Code will just forward the process to the concerned module in the same format as it received the process by putting the process as a message element to the input buffer of the concerned module. The module can form the response in the response format (mentioned below) and send the response back to the corresponding input buffer of the Glue Code. The Glue code will just forward the response to the module which initiated the query. (The Glue Code will get to know about it from the "**.cm"** mentioned at the end of the response received.)

XML File Form:

In case a module needs to send a File as a process parameter, then the module will at first save the file (if the file is not already saved) and find out the absolute path of the file and pass it as a parameter inside the query in the appropriate place…

For example, the Routing Manager of Node A wants to send a file of Routing Table to the Routing Manager of Node B. The Routing Manager will hand over a process of the Communication Manager (in the form of a message) to the Glue Code stating the method and it will also pass the absolute address of the (Routing Table) file as a parameter in the process (message). When the Glue Code gets the process, it simply searches for the module name to which the process belongs to, using the ModuleCheck class. Then the Glue Code puts the process to the input buffer of Communication Manager. When the Communication Manager gets the process from its input buffer, it checks the parameters and then it can locate the File from the file name (along with its' absolute path). The same thing happens when the Communication Manager needs to send a file to the Routing Manager. The Communication Manager will save the file in the hard drive, find the File Name (along with the absolute path) and then it can put the file name along with the absolute path as a

parameter in the process, and send the process as a message to the input buffer of the Glue Code. The Glue Code will see that it is a process and from the ModuleCheck, it will understand that the process belongs to the Routing Manager, and then it will send the process to the input buffer of Routing Manager.

3) Responses:

When the Glue Code does an API call related to a query, the module will just send the response of the query as a reply and nothing else. The Glue Code will form the respective response in the below mentioned format and then send the response to the module which initiated the query. (Glue Code will get to know about which module initiated the query from the "**.cm**" present at the end of the query or the process which has to be preserved even after forming the response.)

All the modules can form the responses in the form of messages and send them to the Glue Code one by one in the following Format:

res.{query_or_process_with_parameters_in_their_correct_sequence}.{response_of_query_or_process_in_appropriate_format}.{acronym_of_the_module_which_sends_the_response}.

Example: Routing Manager can send this following response to the Glue Code:

res.findNextHop(String 6FB23548386CCDB, int 2).details_of_the_next_hop_node.cm

When the Glue Code sees that the message starts with "**res.**", GLUE CODE will understand that this is a response and then the GLUE CODE will check the acronym of the module present at the end of the response and forward the response by putting it at the input buffer of the module which initiated the query.