Introduction

A CALL routine interface has been implemented to provide access to the agent scheduling services. This interface is included with SAS/CONNECT software and can be used within a SAS DATA step or a SAS Macro Facility. This interface provides the user with CALL routines that can run a particular agent.

Dictionary

AGENT_RUN

Run an agent defined to the DOMAIN server.

Syntax

CALL AGENT_RUN(stationId, '_runAgent', agentName, runKey, rc<, description, runLocation, securityInfo, qname>);

<table>
<thead>
<tr>
<th>Where...</th>
<th>Is type...</th>
<th>And represents...</th>
</tr>
</thead>
<tbody>
<tr>
<td>stationId</td>
<td>N</td>
<td>station identifier</td>
</tr>
<tr>
<td>agentName</td>
<td>C</td>
<td>name of agent to run</td>
</tr>
<tr>
<td>runKey</td>
<td>N</td>
<td>run instance key</td>
</tr>
<tr>
<td>rc</td>
<td>N</td>
<td>return code</td>
</tr>
<tr>
<td>description</td>
<td>C</td>
<td>optional text description</td>
</tr>
<tr>
<td>runLocation</td>
<td>C</td>
<td>optional host location to run agent</td>
</tr>
</tbody>
</table>
This function runs an agent that is already defined to the DOMAIN server. There is no way to define an agent through the functional interface; this must be done in an SCL application.

The stationId is the station identifier of a successfully opened station. The function station_open must have been invoked successfully prior to this call.

The agentName parameter is a name of an agent that has already been defined by using the SCL method _defineAgent.

The runKey parameter is set when the agent is run successfully. In an SCL environment, runKey can then be used as an input parameter to _abortAgentRun, _retrieveAgentRunInfo, or _purgeAgentRunInfo. This value is of no subsequent use within the functional environment because there are no other functions supported at this time.

The rc return code identifies whether the agent was run successfully.

The description parameter is an optional user-specified text description of the agent. This string is supplied with run notifications as a presentation context aid.

The runLocation parameter identifies the host and spawner on which the agent should actually run when it executes. The agent may be defined to the DOMAIN server that exists on host A, but the agent may actually be defined to execute on host B.

The securityInfo parameter is the userid.password specification that is needed when the agent actually executes on a given host. This is a required parameter if the target runlocation is a secured environment, otherwise it’s optional.

Note: This may be different than the userid.password specified on the _setDomainInfo SCL method. If different, userid.password is used to communicate with the DOMAIN server when running secured. If the same, it is used to launch the agent on the specified host by using runLocation. △

The qname parameter, if specified, is the name of the domain message queue to which the agent run completion notification is delivered.

Note: All optional parameters must be specified in the order stated by the syntax. △

Example

This program opens the station interface TESTDMM and runs the agent REPORT. Notice that the results are sent to the message queue RQUEUE.

data _null_

    /* open the station */
Call Routine Interface to Agent Services

```plaintext
stid=0;
strate="TESTDMM";
rc=0;
domain="/shr1";
security="user1.pass1";
call station_open(stid, stname, rc, domain, security);

if rc = 0 then put 'Station open successful.';
else do;
    msg = sysmsg();
    put msg;
end;

    /* Run REPORT agent that is already defined */
    /* to the DOMAIN server. Notice that the */
    /* only optional parameter needed is QNAME, */
    /* but the others are specified as NULL */
    /* because the optional parameters are */
    /* positional and must be specified. */
    agName="report";
    runkey=0;
    rc=0;
    desc="";
    runloc="";
    sec="";
    qn="rqueue";
call agent_run(stid, agName, runkey, rc, desc,
                   runloc, sec, qn);

    if rc = 0 then do;
        put 'Agent run successful.';
        put 'Runkey is ' runkey best32.;
    end;
else do;
    msg = sysmsg();
    put msg;
end;

run;
```