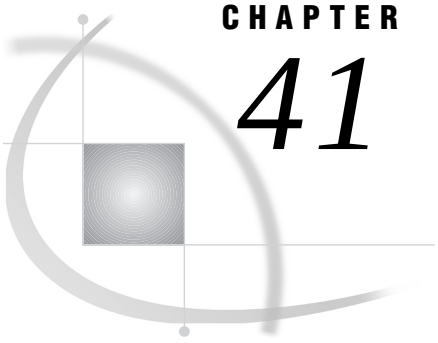

CHAPTER

41

SAS Component Language (SCL) Interface to Agent Services

Introduction

Agent services are provided by the AGENT class.

PARENT:
SASHELP.FSP.OBJECT.CLASS

CLASS:
SASHELP.CONNECT.AGENT.CLASS

Agent Services Methods

The following is a summary of the methods specific to the AGENT class:

_setDomainInfo
Initialize target DOMAIN server context information.

_defineAgent
Define an agent to the DOMAIN server.

_getAgentProperties
Retrieve properties about one or more agents.

_runAgent
Run an agent defined to the DOMAIN server.

_abortAgentRun
Abort an agent run instance.

_retrieveAgentRunInfo
Retrieve the spool for a given agent run instance.

_storeAgentRunInfo
Retrieves and stores the spool for a given agent run instance in an external file or catalog entry.

_purgeAgentRunInfo
Purge the run instance spool for an agent.

_deleteAgent
Delete an agent defined to the DOMAIN server.

The notation that is used to explain the parameter types is

C	Character Type
N	Numeric Type
L	SCL List Type.

Dictionary

_setDomainInfo

Initialize target DOMAIN server context information.

Syntax

CALL SEND(*agentInst*, '_setDomainInfo ', *domainName*, *collectionName*, *rc* <, *securityInfo*, *stationInst*>);

Syntax Description

Where...	Is type...	And represents...
domainName	C	target DOMAIN server location string
collectionName	C	domain logical application partition
rc	N	return code
securityInfo	C	security string (optional)
stationInst	N	previously opened Station instance (optional)

_setDomainInfo method
When invoked on an Agent instance, _setDomainInfo initializes the target DOMAIN server context information. The DOMAIN server information specified here will be used for all subsequent agent class methods that are invoked.

domainName
is the target DOMAIN server location string. It identifies the node and service where the DOMAIN server is executing. For example, the following specification indicates that the DOMAIN server is running on the node MYNODE.XYZ.COM that has a service of DOMSVR:

```
//mynode.xyz.com/domsvr
```

collectionName
is the application collection name. It is a logical partition name (namespace partitions) that allows segregation within a domain. For example, a site may utilize a collection-per-department scheme.

rc
is the return code.

securityInfo
is optional. It is the security string in the form of USERID.PASSWORD. It must be an appropriate userid and password for a login to the system that executes the DOMAIN server. *securityInfo* is needed if the target DOMAIN Server is executing in a secured session.

stationInst
is optional. If it is specified, it should be a previously opened instance. In this case, the domain, the collection, and the security parameters that are specified on the station OPEN are used, and they will take precedence over those same parameters that are specified here.

_defineAgent

Define an agent to the DOMAIN server.

Syntax

CALL SEND(*agentInst*, '_defineAgent ', *agentName*, *rc*
<, *descriptor*, *runLoc*, *securityInfo*, *scheduling*, *notifQueue*, *notifType*, *notifDisp*>);

Syntax Description

Where...	Is type...	And represents...
agentName	C	user-specified name to be assigned to agent
rc	N	return code
descriptor	C	optional text description
runLoc	C	optional host location to run agent
securityInfo	C	optional security information
scheduling	L	optional repetitive run scheduling parameters
notifQueue	C	optional asynchronous notification queue name
notifType	C	optional run notification form
notifDisp	C	optional notification spool disp

_defineAgent method
defines an agent to the DOMAIN server. Any SAS statements currently in the preview buffer are collected and defined to the agent, so that when it runs, the collected statements will execute.

agentName
is a user-specified agent name that uniquely identifies the agent. It must be unique within the application collection.

rc
identifies whether the agent was successfully defined to the DOMAIN server.

All parameters that follow the *rc* parameter are optional and positional. They are defined as follows:

descriptor
is a user-specified text description of the agent.

runLoc
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. For example, the following specification indicates the spawner is running on the node HOSTB.XYZ.COM and has a service name of SPAWNSRV:

```
//hostb.xyz.com/spawnsrv
```

Note: Windows and OS/2 hosts do not require the service name. All other hosts require both the node name and service name. △

securityInfo
is the USERID.PASSWORD specification needed when the agent actually runs (executes) on a given host. This may be different from the USERID.PASSWORD that is specified on the _setDomainInfo method. In that case, *userid.password* is used to communicate with the DOMAIN server when running secured. In this case, *userid.password* is used to launch the agent on the specified host.

scheduling
is an SCL list that contains scheduling information. If no list is specified, the agent is defined to the DOMAIN server but is not scheduled to run. In this situation, the _runAgent method has to be invoked to execute the agent.

The scheduling *parms* list can be specified to identify times and dates on which the agent should run. The bold items that are listed below are supported named-items that can be specified in the list.

Note: Day-of-Week and Month-and-Day scheduling parms are mutually exclusive. Use one form or the other, but not both. △

Day-of-Week Schedule

RUN_DOW
the value 1 represents Sunday and 7 represents Saturday. A single day must be specified by using a single number. Multiple days must be specified by using

comma delimiters (2,4,6 for Monday, Wednesday, Friday). A range must be specified by using a hyphen delimiter (1-7).

Month-and-Day Schedule

RUN_MONTH
the value 1 represents January and 12 represents December. Single, multiple, and range specifications (similar to RUN_DOW) are supported.

RUN_DAY
values range between 1 and 31 to represent the day of the month the agent is scheduled to run. For the agent to execute, the day of the month must be valid.

Time Schedule

RUN_HOUR
indicates hour portion of the agent run launch time (per 24-hour time, 1:00 PM is 13:00).

RUN_MINUTE
indicates minute portion of the agent run launch time.

NotifQueue
identifies the name of the queue in which to send a message when the agent has successfully run. If the queue does not exist, it will be dynamically created. This queue can then be checked periodically by querying the queue to see if there is a message. A message TYPE value of 65539 is reserved and indicates that this message was sent as a result of an agent run completion. The *header* parameter from the query can then be evaluated. The header will contain the following named items:

AGENT_NAME
identifies the name of the agent that ran.

AGENT_RUN_COMPLETION_STATUS
is used to determine whether the agent run was successful.

AGENT_RUN_KEY
is used to retrieve the spool from the run instance.

AGENT_RUN_DATETIME
is the date and time the agent was run.

DESCRIPTOR
is a user-specified text description of the agent.

notifType
indicates the form of the agent run completion notification message. The following values are supported:

COMPLETE
indicates that only completion notification occurs

FULL
indicates that along with completion notification, the run log and listing output spool are included with the notification.

LOG
indicates that along with completion notification, the run log spool is included with the notification.

LIST
indicates that along with completion notification, the listing output spool is included with the notification.

notifDisp
indicates whether the run spool is or is not retained. The following values are supported:

RETAIN
indicates that the run spool is retained after run completion notification. The user must execute the _purgeAgentRunInfo method to delete the spool.

PURGE
indicates that the run spool is deleted subsequent to notification.

Example 1

This program sets a schedule list to run on July 4, at 8:10 p.m.

```
rc = setnitemn(schedule,  7, -1, "RUN_MONTH");
rc = setnitemn(schedule,  4, -1, "RUN_DAY");
rc = setnitemn(schedule, 20, -1, "RUN_HOUR");
rc = setnitemn(schedule, 10, -1, "RUN_MINUTE");
```

Example 2

This program sets a schedule list so that it runs every Monday at 7:30 a.m.

```
rc = setnitemn(schedule,  2, -1, "RUN_DOW");
rc = setnitemn(schedule,  7, -1, "RUN_HOUR");
rc = setnitemn(schedule, 30, -1, "RUN_MINUTE");
```

_getAgentProperties

Retrieve properties about one or more agents.

Syntax

CALL SEND(*agentInst*, '_getAgentProperties ', *agentList*, *rList*, *rc*);

Syntax Description

Where...	Is type...	And represents...
agentList	L	list of agents
rList	L	resulting properties list
rc	N	return code

_getAgentProperties method
retrieves agent properties for a specific agent or multiple agents with the use of the wildcard character (*).

agentList
indicates the targeted agents. This is an SCL list that must contain the named items AGENT_NAME and COLLECTION_NAME. The values of these named items must be character strings that indicate which agent and collections are to be used. You can specify an asterisk (*) as a wildcard character in order to retrieve information about more than one agent. The wildcard character can be specified by itself (*), at the beginning of the value (*value), or at the end of the value (value*).

Note: In addition, the named item USER_NAME can optionally be specified to further subset the request by only returning those agents that are owned by the specified user name. The wildcard character can also be used for the user name specification. △

rList
is the agent(s) properties list returned from the method call. This parameter should be initialized as an empty SCL list. Upon completion of the _getAgentProperties method, *rList* contains a list for each agent that matches the specified criteria. Each list within *rList* contains the following named items:

- AGENT_NAME
- COLLECTION_NAME
- OWNER_NAME

In addition, each list *may* contain one or more of the following named items, depending on how the agent was defined. Only defined properties are returned. For example, if the agent is defined without a description, the named item DESCRIPTOR will not exist.

- DESCRIPTOR
- RUN_LOCATION
- RUN_COMPLETION_QUEUE
- SCHEDULED_TIME
- SCHEDULED_DATE
- RUN_MINUTE
- RUN_HOUR
- RUN_DAY
- RUN_MONTH
- RUN_DOW
- START_DAY
- START_MONTH
- START_YEAR
- STOP_DAY
- STOP_MONTH
- STOP_YEAR

rc
identifies whether the properties of the agent were returned successfully.

Example 1

The following example retrieves properties for all agents in all collections.

```
rc = setnitemc(alist, '*', 'AGENT_NAME');
rc = setnitemc(alist, '*', 'COLLECTION_NAME');
call send(agent, '_getAgentProperties',
          alist, rlist, rc);
```

Example 2

The following example retrieves properties for all agents named EMPLOYEE in all collections that end in TEXAS.

```
rc = setnitemc(alist, 'Employee', 'AGENT_NAME');
rc = setnitemc(alist, '*texas', 'COLLECTION_NAME');
call send(agent, '_getAgentProperties',
          alist, rlist, rc);
```

Example 3

The following example retrieves properties for all agents that start with PROD that are in the CARY collection.

```
rc = setnitemc(alist, 'Prod*', 'AGENT_NAME');
rc = setnitemc(alist, 'Cary', 'COLLECTION_NAME');
call send(agent, '_getAgentProperties',
          alist, rlist, rc);
```

Example 4

The following example retrieves properties for all agents that are owned by the user USER3.

```
rc = setnitemc(alist, '*', 'AGENT_NAME');
rc = setnitemc(alist, '*', 'COLLECTION_NAME');
rc = setnitemc(alist, 'USER3', 'USER_NAME');
call send(agent, '_getAgentProperties',
          alist, rlist, rc);
```

_runAgent

Run an agent defined to the DOMAIN Server.

Syntax

CALL SEND(*agentInst*, '_runAgent ', *agentName*, *runKey*, *rc* <, *descriptor*, *runLoc*, *securityInfo*, *notifQueue*, *notifType*, *notifDisp*>);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent to run
runKey	N	run instance key
rc	N	return code
descriptor	C	optional text description
runLoc	C	optional host location to run agent
securityInfo	C	optional security information
notifQueue	C	optional asynchronous notification queue name
notifType	C	optional run notification form
notifDisp	C	optional notification spool disp

_runAgent method
runs an agent that is defined to the DOMAIN server. If the preview buffer is empty when _runAgent is invoked, the agent must have already been defined by using the _defineAgent method. Otherwise, the SAS statements within the preview buffer are collected and executed when the agent runs.

agentName
is an agent that has been defined to the DOMAIN server using the _defineAgent method.

runKey
is returned from the _runAgent method and can be used to retrieve the run information for this particular run.

rc
parameter identifies whether the launch of the agent run was successful.

All parameters that follow the *rc* parameter are optional and positional. These parameters are

descriptor
is a user-specified text description of the agent.

runLoc
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. For example, the following specification indicates the spawner is running on the node HOSTB.XYZ.COM and has a service name of SPAWNSRV:

```
//hostb.xyz.com/spawnsrv
```

Note: Windows and OS/2 hosts do not require the service name. All other hosts require both the node name and service name. △

securityInfo
is the userid.password specification that is needed when the agent actually runs (executes) on a given host. This may be different from the userid and password that are specified in the _setDomainInfo method. In that case, *userid.password*

communicates with the DOMAIN server when it runs secured. In this case, *userid.password* launches the agent on the specified host by using *runLoc*.

notifQueue
identifies the name of the queue in which to send a message when the agent has successfully executed. This queue can then be checked periodically by querying the queue to see if there is a message. A message TYPE value of 65539 is reserved and indicates that this message was sent as a result of an agent run completion. The header parameter from the query can then be evaluated. The header will contain the following named items:

AGENT_NAME
identifies the name of the agent that ran.

AGENT_RUN_COMPLETION_STATUS
can be used to determine whether the agent run was successful.

AGENT_RUN_KEY
can be used to retrieve the spool from the run instance.

AGENT_RUN_DATETIME
is the date and time the agent was run.

DESCRIPTOR
is a user-specified text description of the agent run.

notifType
indicates the form of the agent run-completion notification message. The following values are supported:

COMPLETE
indicates that only completion notification occurs.

FULL
indicates that along with completion notification, the run log and listing output spool are included with the notification.

LOG
indicates that along with completion notification, the run log spool is included with the notification.

LIST
indicates that along with completion notification, the run listing output spool is included with the notification.

notifDisp
indicates whether the run spool is retained. The following values are supported:

RETAIN
indicates that the run spool is retained after run-completion notification. The user must execute the _purgeAgentRunInfo method to delete the spool.

PURGE
indicates that the run spool is deleted subsequent to notification.

_abortAgentRun

Abort an agent run instance.

Syntax

CALL SEND(*agentInst*, '_abortAgentRun ', *agentName*, *runKey*, *rc*);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent to delete
runKey	N	run instance key
rc	N	return code

_abortAgentRun method
aborts a specific run instance that is executing.

The *agentName* and *runkey* parameters identify the specific run instance of the agent to abort.

agentName
is an agent that has been defined to the DOMAIN server by using the _defineAgent method.

runKey
is returned from the _runAgent method and is used to retrieve the run information for this particular run.

rc
indicates the success or failure of the abort.

_retrieveAgentRunInfo

Retrieves the spool for a given agent run instance.

Syntax

CALL SEND(*agentInst*, '_retrieveAgentRunInfo ',
agentName, *runKey*, *dateTime*, *cc*, *rc*
<, *logSpool*, *listSpool*>);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent
runKey	N	run instance key
dateTime	N	date-time stamp

Where...	Is type...	And represents...
cc	N	completion code
rc	N	return code
logSpool	N	optional log spool model object id
listSpool	N	optional list spool model object id

_retrieveAgentRunInfo method
retrieves the log and output spool for a given agent run, which is identified by the *agentName* and *runKey* parameters. The output can be displayed in a FRAME text viewer.

agentName
is an agent that has been defined to the DOMAIN server using the _defineAgent method.

runKey
is returned from the _runAgent method and can be used to retrieve the run information for this particular run.

dateTime
is a return parameter that holds the date/time stamp for the run instance.

cc
is a return parameter that holds the completion code; if *cc* is 0, the agent run executed successfully.

rc
parameter indicates whether the _retrieveAgentRunInfo method was invoked successfully.

All parameters after *rc* are optional and positional. These parameters are

logSpool
is the log spool model object id that identifies the model in which to display the log spool that is retrieved. The SASHELP.CONNECT.RUNSPOOL class can be used to present the run instance log spool.

listSpool
is the listing output spool model object id that identifies the model in which to display the listing output spool that is retrieved. The SASHELP.CONNECT.RUNSPOOL class can be used to present the run instance listing output spool.

Note: Both log and listing output may be directed to the same spool instance or to different run spools. Either or both may be omitted as well. The run spool instances may be displayed with the FRAME text viewer. △

Example

This example retrieves the agent run information and displays the log and listing output run spool in a FRAME text viewer instance named VIEWER_I.

```
model_c = loadclass('sashelp.connect.runspool.class');
model_i = instance(model_c);

call send(agentInst, '_retrieveAgentRunInfo',
```

```
            agent_name, runkey, rundt, runcc, rc,
            model_i, model_i);

  call notify('viewer_i', '_ATTACH_', model_i);
```

_storeAgentRunInfo

Retrieves and stores the spool for a given agent run instance in an external file or catalog entry.

Syntax

CALL SEND(*agentInst*, '_storeAgentRunInfo ',
agentName, runKey, dateTime, cc, type, logSpec, listSpec, rc);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent
runKey	N	run instance key
dateTime	N	date/time stamp
cc	N	completion code
type	C	type of storage file
logSpec	C	external file or catalog in which to store the log
listSpec	C	external file or catalog in which to store the listing output
rc	N	return code

_storeAgentRunInfo method
retrieves the log and listing output spool for a given agent run which is identified by the *agentName* and *runKey* parameters and stores it in either an external file or catalog entry.

agentName
is an agent that has been defined to the DOMAIN server using the _defineAgent method.

runKey
is returned from the _runAgent method and is used to retrieve the run information for this particular run.

dateTime
is a return parameter that holds the date/time stamp for the run instance.

cc
is a return parameter that holds the completion code. If 0, the agent run executed successfully.

type
parameter indicates the type of file in which the agent run information will be stored. This parameter must have a value of either CATALOG or EXTERNAL.

logSpec
listSpec
parameters indicate the file(s) in which to store the log and listing output from the agent run. These parameters will vary based on the value of *type*:

- If *type* is CATALOG, *logSpec* and *listSpec* must be a valid catalog names that represent the catalog entry. It must be in the form of

  ```
  libname.memname.entryname
  ```

  entrytype does not have to be specified because the log information is stored in an **entrytype** of AGENTLOG and the output is stored in an entrytype of AGENTOUT.
- If *type* is EXTERNAL, *logSpec* and *listSpec* must be valid filerefs. When an external file is used, the user has the flexibility to decide whether to store the log and listing output in the same file or in separate files. If they are identical, the log and listing output are stored in the same file. If not, the log and listing output are stored separately.

rc
indicates whether the _storeAgentRunInfo method was invoked successfully.

Example

This example stores the agent run info in the external file EXTREF, which is defined by the fileref.

```
ftype = 'EXTERNAL';
fref  = 'extref';
rc = filename('extref', '/tmp/agent.txt');
call send(agentInst, '_storeAgentRunInfo',
          agent_name, runkey, rundt, runcc,
          ftype, fref, rc);
```

_purgeAgentRunInfo

Purge the run instance spool of the agent.

Syntax

CALL SEND(*agentInst*, '_purgeAgentRunInfo ',
agentName, *runKey*, *rc*);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent to purge
runKey	N	run instance key
rc	N	return code

_purgeAgentRunInfo method
purges the spool from a specific run instance that is identified by the *agentName* and the *runKey* parameters.

agentName
is an agent that has been defined to the DOMAIN server by using the _defineAgent method.

runKey
is returned from the _runAgent method and is used to retrieve the run information for this particular run.

rc
indicates the success or failure of the purge.

_deleteAgent

Delete an agent defined to the DOMAIN Server.

Syntax

CALL SEND(*agentInst*, '_deleteAgent ', *agentName*, *rc*);

Syntax Description

Where...	Is type...	And represents...
agentName	C	name of agent to delete
rc	N	return code

_deleteAgent method
deletes an agent defined to a particular DOMAIN server

agentName
is an agent that has been defined to the DOMAIN server by using the _defineAgent method.

rc
indicates the success or failure of the deletion.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

ISBN 1–58025–477–2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999