CHAPTER 21

SAS Component Language (SCL) Interface to Remote Objecting

Remote Objecting Methods

The following methods that are specific to the ROBJECT class are described in this section.

_createRemoteInstance

_beginMethod

_addMethodArgC

_addMethodArgN

_addMethodArgL

_invokeMethod

_destroyRemoteInstance

Note: Notation that is used to explain the parameter types is as follows:

C	Character Type
N	Numeric Type
L	SCL List Type

△

_createRemoteInstance

Creates instance in remote SAS session.

Syntax

CALL SEND(*robjInst*, '_createRemoteInstance',
remote_name, *class_name*, *rc*);

Where...	Is type...	And represents...
remote_name	C	remote destination
class_name	C	fully-qualified class name
rc	N	return code

_createRemoteInstance
creates an instance of the specified class in the remote environment. The user is responsible for obtaining the instance of the ROBJECT class, and then sending it the _createRemoteInstance method. After the _createRemoteInstance method has successfully completed, methods can be invoked using the remote instance.

remote_name
designates the remote destination in which to create the instance. It is the concatenation of the keyword "remote", a double-slashed delimiter, and then the REMOTE= value that was supplied in the SIGNON statement. For example, if OAK is the node signed onto, then the remote_name would be, **"remote//oak"**.

class_name
is the fully qualified class name that is used to create a remote instance (that is, library.catalog.classname).

rc
is a return code that indicates success or failure. A value of zero indicates success. A non-zero value indicates failure.

Example

This example creates a remote instance of the class **sashelp.fsp.object** by first signing on to the remote host OAK and then issuing the _createRemoteInstance method.

```
filename rlink 'tcpunix.scr';
signon oak;

robj = loadclass('sashelp.connect.robject.class');
robjInst = instance(robj);
call send(robjInst,
          "_createRemoteInstance",
          "remote//oak",
          "sashelp.fsp.object",
          rc);
```

_beginMethod

Begins defining method to invoke on remote instance.

Syntax

CALL SEND(*robjInst*, '_beginMethod', *method_name*, *rc*);

Where...	Is type...	And represents...
method_name	C	name of method to invoke
rc	N	return code

_beginMethod
is invoked on an instance of the ROBJECT class to begin defining a method. This merely begins the definition stage of the method to invoke on the remote instance that is created by the _createRemoteInstance method.

method_name
is the name of the method to begin defining. Again, method_name is not actually invoked on the remote instance until _invokeMethod is executed; this is merely the definition phase.

rc
is a return code that indicates success or failure. A value of zero indicates success. A non-zero value indicates failure.

_addMethodArgC

Adds character parameter to method call.

Syntax

CALL SEND(*robjInst*, '_addMethodArgC', *value*, *mode*, *rc*, <*name*>);

Where...	Is type...	And represents...
value	C	character parameter to be passed to method
mode	C	mode of parameter
rc	N	return code
name	C	optional name to be associated with this character parameter

_addMethodArgC
builds the method call that is invoked on the remote instance. It enables character parameters to be added to the method definition.

value
is the actual character parameter that is passed to the remote method invocation.

mode
indicates the mode of this parameter:

I = input parameter

O = output parameter

U = update parameter.

rc
is a return code that indicates success or failure. A value of zero indicates success. A non-zero value indicates failure.

name
may be specified to associate a name with this character parameter.

_addMethodArgN

Adds numeric parameter to method call.

Syntax

CALL SEND(*robjInst*, '_addMethodArgN', *value*, *mode*, *rc*, <*name*>);

Where...	Is type...	And represents...
value	N	numeric parameter to be passed to method
mode	C	mode of parameter
rc	N	return code
name	C	optional name to be associated with this numeric parameter

_addMethodArgN
builds the method call that is invoked on the remote instance. It enables numeric parameters to be added to the method definition.

value
is the actual numeric parameter that is passed to the remote method invocation.

mode
indicates the mode of this parameter:

I = input parameter

O = output parameter

U = update parameter

rc
is a return code that indicates success or failure. A zero value indicates success. A non-zero value indicates failure.

name
may be specified to associate a name with this numeric parameter.

_addMethodArgL

Adds SCL list parameter to method call.

Syntax

CALL SEND(*robjInst*, '_addMethodArgL', *value*, *mode*, *rc*, <*name*>);

Where...	Is type...	And represents...
value	L	SCL list parameter to be passed to method
mode	C	mode of parameter
rc	N	return code
name	C	optional name to be associated with this SCL list parameter

_addMethodArgL
builds the method call that is invoked on the remote instance. It enables SCL list parameters to be added to the method definition.

value
is the actual SCL list parameter that is passed to the remote method invocation.

mode
indicates the mode of this parameter:

I = input parameter

O = output parameter

U = update parameter.

rc
is a return code that indicates success or failure. A zero value indicates success. A non-zero value indicates failure.

name
may be specified to associate a name with this SCL list parameter.

_invokeMethod

Invokes method on remote instance.

Syntax

CALL SEND(*robjInst*, '_invokeMethod', *return_list*, *rc*);

Where...	Is type...	And represents...
return_list	L	SCL list of output/update parameters
rc	N	return code

_invokeMethod
invokes the method on the remote instance and passes to it all parameters that were defined by using the add argument methods.

return_list
is an SCL list that contains any output or update parameters that are returned by the remote method invocation. *return_list* parameter always contains the named item _MRC, which is the return code from the remote method invocation. _MRC indicates whether the remote method call was invoked without errors (syntax error or incorrect number of parms, and so forth). It does not indicate how the method ran. It only indicates whether it was invoked successfully. If _MRC is zero, the user can process *return_list* further to evaluate return parameters that might indicate how the method ran.

rc
is a return code that indicates success or failure. A zero value indicates success. A non-zero value indicates failure.

Example

This example illustrates the code to use for checking _MRC and displays all returned parameters in the return list.

```
rlist = makelist();
call send(robj, '_invokeMethod', rlist,rc);
if (rc eq 0) then do;

      /* get named item _MRC to determine if  */
      /* remote method invoked without syntax */
      /* error or wrong number of parms       */
   mrc = getnitemn(rlist, '_mrc', 1, 1, 0);
   if (mrc eq 0) then do;

         /* retrieve returned parms (and */
         /* optionally a name if it is a */
         /* named item) and dump to log  */
      do i = 1 to listlen(rlist);
         select( itemtype(rlist, i) );

            /* character parameter returned */
         when('C')
```

```
            do;
               cparm = getitemc(rlist, i);
               name='';
               name = nameitem(rlist, i);
               put 'Returned character parm is '
                  cparm name;
            end;

            /* numeric parameter returned */
         when('N')
            do;
               nparm = getitemn(rlist, i);
               name='';
               name = nameitem(rlist, i);
               put 'Returned numeric parm is'
                  nparm name;
            end;

            /* list parameter returned */
         when('L')
            do;
               lparm = getiteml(rlist, i);
               name='';
               name = nameitem(rlist, i);
               put 'Returned list parm is'
                  lparm name;
            end;
      end;
   end; /* end if mrc eq 0 */
end; /* end if rc eq 0 */
```

_destroyRemoteInstance

Destroys remote instance.

Syntax

CALL SEND(*robjInst*, '_destroyRemoteInstance', *rc*);

Where...	Is type...	And represents...
rc	N	return code

_destroyRemoteInstance
terminates the remote instance and frees any associated resources. The ROBJECT instance still exists but another remote instance must be instantiated before it is useful.

rc
is a return code that indicates success or failure. A zero value indicates success. A non-zero value indicates failure.

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