CHAPTER

22

Examples That Use Remote Objecting

Example 1 - Create a Remote Instance

This example creates a remote instance of the SASHELP.FSP.DATAST_M class. After the remote instance is created, the methods _SET_DATASET_ and _GET_DATASET_NAME_ are invoked remotely, and the results are returned for local processing.

```
main:
   r_string = '';
   c_string = '';
   rc=0;

      /***********************************/
      /* Load class ROBJECT then create an*/
      /* instance of that class locally.  */
      /***********************************/
   remote_c =
      loadclass('sashelp.connect.robject');
   rid = instance(remote_c);

      /***********************************/
      /* Define the remote host and the   */
      /* location of the class on which   */
      /* method would be used.            */
      /***********************************/
   remval = optgetc('REMOTE');
   c_string = 'sashelp.fsp.datast_m';
   r_string = 'remote//' || remval;

      /***********************************/
      /* Create remote instance of the    */
      /* class.                           */
      /***********************************/
   call send(rid, '_createRemoteInstance',
             r_string, c_string, rc);

      /***********************************/
```

```
   /* Begin with method "_SET_DATASET_"*/
   /**********************************/
call send(rid, '_beginMethod',
          "_SET_DATASET_", rc);
put ' ';
put '_beginMethod of _SET_DATASET_
     was successful ... ';

   /**********************************/
   /* Begin sending information to the */
   /* method. Send name of class;      */
   /* mode INPUT. Be sure the data     */
   /* set itself exists.               */
   /**********************************/
dsname = 'work.a';
c_mode = "I";
c_name = "remote dataset name";
put ' ';
put "submitting the name of the remote
     dataset " dsname c_mode c_name;
call send(rid, '_addMethodArgC', dsname,
          c_mode, rc, c_name);

   /**********************************/
   /* After all required information   */
   /* has been sent to the method,     */
   /* invoke _SET_DATASET_.            */
   /**********************************/
put ' ';
put 'Invoking _SET_DATASET_ method...';
r_list = makelist();
call send(rid, '_invokeMethod', r_list,rc);
put 'Invoke method _SET_DATASET_
     was successful ... ';

   /**********************************/
   /* Begin method                     */
   /* "_GET_DATASET_NAME_".            */
   /**********************************/
call send(rid, '_beginMethod',
          "_GET_DATASET_NAME_", rc);
put ' ';
put '_beginMethod of _GET_DATASET_NAME_
        was successful ... ';

   /**********************************/
   /* Begin sending information to the */
   /* method; mode OUTPUT because      */
   /* results are expected now.        */
   /**********************************/
dsname = '';
c_mode = "O";
c_name = "return the dataset name";
put "expecting a result variable back "
```

```
   dsname c_mode c_name;
call send(rid, '_addMethodArgC', dsname,
          c_mode, rc, c_name);

   /***********************************/
   /* After all required information   */
   /* has been sent to the method,     */
   /* invoke _GET_DATASET_NAME_.       */
   /***********************************/
put ' ';
put 'Invoking _GET_DATASET_NAME_ method...';
r_list = makelist();
call send(rid, '_invokeMethod', r_list,rc);
put 'Invoke method was successful...';

if (rc eq 0) then
   do;
      mrc = getnitemn(r_list, '_mrc',
                      1, 1, 0);
      put ' ';
      put "method return code= " mrc;
      put "now we need to fetch the result
           from the result list";

      /*********************************/
      /* After the method is invoked,   */
      /* a list is created that holds   */
      /* the result. Fetch the results  */
      /* from the list and display them.*/
      /*********************************/
     if (mrc = 0) then
        do index = 1 to listlen(r_list);
           select(itemtype(r_list, index));
           when('C')
              do;
                 rem_varc =
                    getitemc(r_list, index);
                 c_name='';
                 c_name =
                    nameitem(r_list, index);
                 put '_GET_DATASET_NAME_
                      when character: '
                      rem_varc c_name;
              end;
           when('N')
              do;
                 rem_varn =
                    getitemn(r_list, index);
                 c_name='';
                 c_name =
                    nameitem(r_list, index);
                 put '_GET_DATASET_NAME_
                      when numeric: '
                      rem_varn c_name;
```

```
                end;
             when('L')
                do;
                   rem_varl =
                      getiteml(r_list, index);
                   c_name='';
                   c_name =
                      nameitem(r_list, index);
                   put '_GET_DATASET_NAME_
                        when list: '
                        rem_varl c_name;
               end;
            end;
         end;
      end;

      /************************************/
      /* For clean-up purposes, destroy the */
      /* remote instance.                   */
      /************************************/
   put ' ';
   put 'Destroying remote instance ... ';
   call send(rid, '_destroyRemoteInstance',rc);
   put 'remote instance destroyed successful';
return;
```

Example 2 - Gather Class Information

In this example, information is gathered remotely about a class. _GET_METHODS_ is used to return the names of all the methods that are defined for a specific class.

```
   /************************************/
   /* Define rid (remote ID).          */
   /************************************/
init:
   rid = 0;
return;

main:
   r_string = '';
   c_string = '';
   rc=0;

   remote_c = loadclass('sashelp.connect.robject');
   rid = instance(remote_c);

      /**********************************/
      /* Define the remote host and     */
      /* location of the class.         */
      /**********************************/
   remval = 'MARS';
   classname =
      'permdata.roclass.ro2tst.class';
```

```
r_string = 'remote//' || remval;

   /***********************************/
   /* Create remote instance of class. */
   /***********************************/
call send(rid, '_createRemoteInstance',
          r_string, classname, rc);

   /***********************************/
   /* Begin method "SEND_METHOD_NUM".  */
   /***********************************/
call send(rid, '_beginMethod',
          "SEND_METHOD_NUM", rc);


/***************************************/
/* Method "SEND_METHOD_CHR" accepts the  */
/* following parameters:                 */
/*   (1) the first parameter must be     */
/*         the class name,               */
/*   (2) the second, is the method name  */
/*   (3) the third, is the result of the */
/*       call that is being returned.    */
/***************************************/

/***************************************/
/* Send name of class; mode INPUT.     */
/***************************************/
cls = 'permdata.roclass.ro2tst';
c_mode = "I";
c_name = "remote class name";
put "submitting the name of remote class "
   cls c_mode c_name;

call send(rid, '_addMethodArgC', cls,
          c_mode, rc, c_name);

   /***********************************/
   /* Send name of an SCL method to    */
   /* be used on the remote side;      */
   /* mode INPUT.                      */
   /***********************************/
mth = '_get_methods_';
c_mode = "I";
c_name = "remote method name";
put "submitting the method to remote side "
   mth c_mode c_name;

call send(rid, '_addMethodArgC', mth,
          c_mode, rc, c_name);

   /***********************************/
   /* Update the RESULT parameter to   */
   /* hold new results; mode OUTPUT.   */
```

```
   /*********************************/
result=makelist();
c_mode = "O";
c_name = "result of the remote method";
put "getting the result from remote side "
   result c_mode c_name;

call send(rid, '_addMethodArgL', result,
          c_mode, rc, c_name);

   /*************************************/
   /* After all the information has       */
   /* been sent to the remote method,     */
   /* use INVOKE_METHOD "SEND_METHOD_NUM" */
   /*************************************/
put ' ';
put 'Invoking SEND_METHOD_NUM method...';
r_list = makelist();

call send(rid, '_invokeMethod',
          r_list, rc);

if (rc eq 0) then
   do;
      mrc = getnitemn(r_list, '_mrc',
                      1, 1, 0);
      put "method return code= " mrc;

         /*******************************/
         /* After the method is invoked,*/
         /* a list is created that holds*/
         /* the result.                 */
         /* Fetch the results from that */
         /* list and display them to    */
         /* the log.                    */
         /*******************************/
      if (mrc = 0) then
        do index = 1 to listlen(r_list);
          select(itemtype(r_list, index));
          when('C')
            do;
              rem_varc = getitemc(r_list,
                                  index);
              c_name = '';
              c_name = nameitem(r_list,
                                index);
              put 'SEND_METHOD_NUM for
                   character : '
                   rem_varc c_name;
            end;
          when('N')
            do;
              rem_varn = getitemn(r_list,
                                  index);
```

```
                c_name = '';
                c_name = nameitem(r_list,
                                  index);
                put 'SEND_METHOD_NUM for
                     numeric : '
                     rem_varn c_name;
              end;
            when('L')
              do;
                rem_varl = getiteml(r_list,
                                    index);
                 c_name = '';
                 c_name = nameitem(r_list,
                                   index);
                 put 'SEND_METHOD_NUM for
                      list : '
                      rem_varl c_name;
              end;
            end;
          end;
      end;

      /***********************************/
      /* _GET_METHODS_ returns a         */
      /* list; so the result is a list   */
      /* within r_list (the list that    */
      /* SEND_METHOD_NUM created).  The  */
      /* remote list needs to be captured,*/
      /* and its contents displayed.     */
      /***********************************/
    put ' ';
      if (mrc = 0) then
         do index = 1 to listlen(rem_varl);
           select(itemtype(rem_varl, index));
           when('C')
             do;
               varc = getitemc(rem_varl,
                               index);
               put '**** character RESULT
                    **** : ' varc ;
             end;
           when('N')
             do;
               varn = getitemn(rem_varl,
                               index);
               put '**** numeric RESULT
                    **** : ' varn ;
             end;
           when('L')
             do;
               varl = getiteml(rem_varl,
                               index);
               put '**** list RESULT
                    **** : ' varl;
```

```
            end;
          end;
        end;

      /***********************************/
      /* For clean-up purposes, destroy  */
      /* the remote instance.            */
      /***********************************/
   put 'Destroying remote instance ... ';
   call send(rid, '_destroyRemoteInstance',
            rc);
return;
```

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