CHAPTER

19 Examples of Data Transfer Services (DTS)

Example 1. DTS: Transferring Data by Using WHERE Statements

Purpose

The UPLOAD and DOWNLOAD procedures process WHERE statements and the WHERE= data set option when you transfer a single SAS data set. The transferred data set contains only the observations that meet the WHERE condition.

Program

```
proc upload data=school out=kindergarten;
   where class='K';
run;
```

Example 2. DTS: Transferring Specific Member Types by Using SELECT or EXCLUDE Statements

Purpose

If you include the INLIB= and OUTLIB= options in the PROC UPLOAD or PROC DOWNLOAD statements, you can specify which member types to transfer by using the MEMTYPE= option in one of the following statements:

- PROC UPLOAD
- PROC DOWNLOAD
- SELECT
- EXCLUDE.

Valid values for the MEMTYPE= option are DATA, CATALOG (or CAT), MDDB view, FDB, DBDB, and ALL. If you use this option in the EXCLUDE statement, you can specify only one value. If you use this option in the PROC UPLOAD or the PROC DOWNLOAD statement, you can specify a list of MEMTYPE values enclosed in parenthesis.

Programs

Example 2.1: Using the MEMTYPE= Option in the PROC UPLOAD Statement

This example uploads all catalogs and data sets that are in the library THIS on the local host and stores them in the library THAT on the remote host.

```
proc upload inlib=this outlib=that
   memtype=(data catalog);
```

Example 2.2: Using the MEMTYPE= Option in the EXCLUDE Statement

This example uploads all catalogs and data sets except the data sets that are named Z4, Z5, Z6, and Z7 that are in the library LOCLIB on the local host and stores them in the library REMLIB on the remote host:

```
proc upload inlib=loclib outlib=remlib mt=all;
   exclude z4-z7 / memtype=data;
run;
```

Example 2.3: Using the MEMTYPE= Option in the SELECT Statement

This example downloads the catalogs NAMES and SALARY and the data set MEDIA in the data library REMLIB on the remote host and stores them in the library LOCLIB on the local host:

```
proc download inlib=remlib outlib=loclib;
   select names salary media(mt=data) / memtype=cat;
run;
```

Example 3. DTS: Transferring Specific Catalog Entry Types

Purpose

When you include the INCAT= and OUTCAT= options in the PROC UPLOAD or PROC DOWNLOAD statement, you can specify which entry types to transfer by using the ENTRYTYPE= option in one of the following statements:

- PROC UPLOAD
- PROC DOWNLOAD
- SELECT
- EXCLUDE.

If you omit the ENTRYTYPE= option and also omit the SELECT and EXCLUDE statements, all catalog entries are transferred.

Programs

Example 3.1: Using the ENTRYTYPE= Option in the PROC UPLOAD Statement

This example uploads all SLIST catalog entries from the CAT catalog in the library LOCLIB on the local host and stores them in the catalog UPCAT in the library REMLIB on the remote host:

```
proc upload incat=loclib.cat
   outcat=remlib.upcat entrytype=slist;
run;
```

Example 3.2: Using the ENTRYTYPE= Option in the EXCLUDE Statement in PROC DOWNLOAD

This example downloads all catalog entries except the format entries XYZ and GRADES, which are in the catalog REMOTE.MAIN_FORMATS on the remote host and stores them in the catalog LOCAL.SECONDARY_FORMATS on the local host:

```
proc download incat=remote.main_formats
   outcat=local.secondary_fomats;
   exclude xyz grades / entrytype=format;
run;
```

Example 3.3: Using the ENTRYTYPE= Option in the SELECT Statement in PROC UPLOAD

If the default library is WORK, this example uploads the FORMAT catalog entries XYZ and ABC, the INFMT catalog entry GRADES, and the SCL entries A and B, which are in the WORK.LOCFMT catalog on the local host and stores them in the WORK.REMFMT catalog on the remote host:

```
proc upload incat=locfmt outcat=remfmt;
   select xyz.format grades
      abc (et=format) / et=infmt;
   select a b / et=scl;
run;
```

Example 3.4: Using the ENTRYTYPE= Option in Two SELECT Statements

This example maintains the original ordering and grouping when transferring catalog entries that contain graphics output. Assume that you have a catalog named FINANCE that has two entries that contain graphics output, INCOME and EXPENSE. You want to download the two catalog entries that contain graphics output in the order in which they are stored on the remote host; that is, you want INCOME to appear before EXPENSE, not alphabetically as the DOWNLOAD procedure would normally transfer them.

In addition, you have some catalog entries that are grouped by the name GROUP1, and you want to preserve the grouping when the entries are downloaded. Remote submit the following program to transfer these entries in the order that you specify in the first SELECT statement and in the group that you specify in the second SELECT statement:

```
proc download incat=rhost.finance
   outcat=lhost.finance;
   select income expense et=grseg;
   select group1;
run;
```

Example 3.5: Using Long Member Names in Catalog Transfers

This example uses PROC UPLOAD to transfer catalogs by using the INCAT= and/or OUTCAT= options:

```
rsubmit;
   proc upload
      incat=loclib.monthlysalary
      outcat=monthlyupdate;
   run;
   proc upload
      incat=loclib.employeedata
      outcat=remlib.cat;
   run;

   proc upload incat=sasuser.base
      outcat = remlib.basecatalog;
   run;

endrsubmit;
```

Example 4. DTS: Transferring Generations of SAS Data Sets

Purpose

Generation data sets are historical versions of SAS data sets, SAS views, and SAS/ACCESS files. They enable you to keep a historical record of the changes that you make to these files. There are two data set options that are useful when manipulating generations of SAS data sets: maximum number of generations (GENMAX) and generation number (GENNUM). GENMAX indicates how many generations to keep, and GENNUM is used to access a specific version of a generation group.

SAS/CONNECT transfers generations of SAS data sets by default during library transfers. The base data set, as well as all of its historical versions, are transferred.

If the user does not want all generations to be transfered, single data set transfers should be used. With single data set transfers, only the specified data set is transferred.

Programs

Example 4.1: Using LIBRARY Transfers to Transfer Data Set Generations

This example transfers the local data set LOCAL.SALES as well as its generations to the remote library REMOTE. If the data set SALES already exists in the output library, the base and all existing generations will be deleted and replaced by those that are uploaded.

```
data local.sales(genmax=3);
   input store sales95 sales96 sales97;
   datalines;
  1   221325.85   214664.02   212644.60
  2   134511.96   159369.47   317808.48
  3   321662.42   244789.33   236782.59
 ;
run;

data local.sales;
   input store sales95 sales96 sales97;
   datalines;
  1   251325.25   217662.16   222614.60
  2   144512.11   179369.47   327808.48
  3   329682.43   249989.93   256782.59
 ;
run;

data local.sales;
   input store sales95 sales96 sales97;
   datalines;
  1   261325.33   218862.16   222614.60
  2   145012.11   189339.47   328708.71
  3   330682.46   259919.92   258722.52
 ;
run;
```

```
   /* PROC DATASETS will show that the   */
   /* base data set as well as two       */
   /* generations exist in the library.  */
proc datasets lib=local;
quit;

rsubmit;
   proc upload in=local out=remote cstatus=no;
   run;
endrsubmit;
```

Example 4.2: Using a SELECT Statement to Transfer Generations

Specific generations cannot be specified in the SELECT or the EXCLUDE statements for library transfers. When the SELECT statement is specified for the library transfer, the selected base data set as well as all of its historical versions will be transferred. Similarly, when the EXCLUDE statement is specified for the library transfer, the selected base data set as well as all of its historical versions will be excluded from the transfer.

In the following example, the data set LOCAL.SALES as well as all of its generations will be uploaded.

```
rsubmit;
   proc upload in=local out=remote cstatus=no;
      select sales (mt=data);
   run;
endrsubmit;
```

Example 4.3: Inheriting Generation Specific Attributes

During library transfers and single data set transfers when OUT= is not specified, data set attributes are inherited in the output data set. In Version 7 or Version 8, the maximum number of generations will be a new inherited attribute. In addition, the next generation number attribute is inherited ONLY when a library transfer occurs. This attribute is only inherited when the generations are actually transferred, and therefore it is NOT inherited for any single data set transfers. In the following example, both the maximum number of generations and the next generation number attributes are inherited in the output data set because this is a library transfer.

```
rsubmit;
   proc download in=remote out=local;
      select sales(mt=data);
   run;
endrsubmit;
```

In the following example, only the maximum number of generations attribute is inherited. The next generation number attribute is not inherited because this is a single data set transfer, and therefore no generations are transferred.

```
rsubmit;
   proc download data=remote.sales;
   run;
endrsubmit;
```

Example 4.4: Transferring Single Data Sets

A specific generation can be transferred by specifying the GENNUM= data set option for a single data set transfer. In the following example, a specific historical version is updated by specifying GENNUM=1.

```
rsubmit;
   proc upload data=local.sales(gennum=1);
   run;
endrsubmit;
```

Example 5. DTS: Transferring Long Member Names

Purpose

SAS/CONNECT supports the transfer of long member names for single data set transfers, as long as the host supports long member names. This example uses PROC UPLOAD and PROC DOWNLOAD to transfer a data set and a catalog that have long member names:

Program

```
rsubmit;
   proc upload in=work out=sasuser;
      select longdatasetname(mt=data)
      cat longcatalogname/mt=cat;
   run;

   data x.sas_institute_employee_data;
     set empdata;
   run;

   proc download inlib=x outlib=work;
   run;
endrsubmit;
```

Example 6. DTS: Transferring Data by Using Data Set Options and Attributes

Purpose

PROC UPLOAD and PROC DOWNLOAD permit you to specify SAS data set options in the DATA= and OUT= options. Note that SAS data set options are not supported when using the INLIB= and OUTLIB= options, even when you upload only data sets.

The data set options must be associated with a specific SAS data set, so they must be used in the DATA= or OUT= options. There are additional restrictions described in

Chapter 17, "The UPLOAD Procedure," on page 107 and Chapter 18, "The DOWNLOAD Procedure," on page 129.

This example illustrates using the DATA= option and the INDEX=NO option. It also shows the use of the RENAME= and DROP= SAS data set options. Note that because no OUT= option is specified, the transferred data set inherits all of the characteristics of the input data set except for the index (because the INDEX=NO option is specified).

Program

```
proc download data=survey
   (rename(r=response) drop=comments)
   index=no;
run;
```

Example 7. DTS: Transferring Data Set Integrity Constraints

PROC UPLOAD and PROC DOWNLOAD in SAS/CONNECT permit a transferred SAS data set to inherit the characteristics of the input data set. If the OUT= option is omitted when transferring a specific SAS data set, then the transferred data set inherits the characteristics of the input data set. A transferred data set also inherits the characteristics of the input data set if it is part of a library transfer (See the INLIB= and OUTLIB= options for PROC UPLOAD and PROC DOWNLOAD).

A new potential characteristic of a SAS data set known as integrity constraints has been added for SAS Version 7 or Version 8. Integrity constraints are a set of data validation rules that preserve the consistency and correctness of the stored data. These rules are defined by the applications programmer and are enforced by SAS for each request to modify the data.

Purpose

PROC UPLOAD and PROC DOWNLOAD have been modified for Version 7 or Version 8 to enable the transfer of integrity constraints that are defined on a data set. As with other data set characteristics, integrity constraints are inherited by a transferred data set under the conditions stated above. The only exception to this will be if the input file has an index defined and the user specifies the INDEX=NO option, then any integrity constraints that are defined for the input file will not be inherited. Also, referential integrity constraint types are never transferred.

Programs

Example 7.1: Omitting the OUT= Option from the PROC DOWNLOAD Statement

This example downloads the SAS data set REM in the library WORK on the remote host to the library WORK on the local host. Any non-referential integrity constraints defined for the input data set are inherited by the output data set.

```
proc download data=rem;
```

Example 7.2: Using the DROP= Option in the PROC UPLOAD Statement

This example uploads the SAS data set LOC in the library WORK on the local host to the library WORK on the remote host. The variable ONE is dropped from the output data set. Any non-referential integrity constraints that are defined for the input data set that do not include the variable ONE are inherited by the output data set.

```
proc upload data=loc(drop=one);
```

Example 7.3: Using the INLIB= Option in the PROC UPLOAD Statement

This example uploads all SAS data sets in the library SASUSER on the local host and stores them in the library WORK on the remote host. Any non-referential integrity constraints that are defined for each of the input data sets are inherited by the corresponding output data set.

```
proc upload inlib=sasuser outlib=work;
```

Example 7.4: Using the INDEX=NO Option in the PROC DOWNLOAD Statement

This example downloads the SAS data set STUDENTS in the library WORK on the remote host to the library WORK on the local host. Any non-referential integrity constraints defined for the input data set are inherited by the output data set unless there are indexes defined on the input data set, then no integrity constraints are defined for the output data set.

```
proc download data=students index=no;
```

Example 8. DTS: Transferring Numerics by Using the EXTENDSN= and V6TRANSPORT Options

Purpose

For releases prior to Version 7, when transferring short numerics (length less than 8) the length of these numerics is automatically increased to preserve precision. In Version 7 or Version 8, the length of these numerics are increased by default unless the V6TRANSPORT option is specified. Using the V6TRANSPORT and EXTENDSN= options in PROC UPLOAD and PROC DOWNLOAD statements, you have the choice of whether or not to promote the length of numerics.

Programs

Example 8.1: Using the EXTENDSN= and V6TRANSPORT Options in the PROC UPLOAD Statement

This example uploads the data set A in the directory WORK on the local host to the directory REMOTE on the remote host. The V6TRANSPORT option causes the short numerics to be promoted; therefore EXTENDSN=NO must be specified to override this default, so that numerics will not be promoted.

```
proc upload data=a out=remote
   v6transport extendsn=no;
run;
```

Example 8.2: Using the EXTENDSN= Option in the PROC DOWNLOAD Statement

This example downloads the catalog SCAT in the directory REMOTE on the remote host to the directory WORK on the local host. By default, catalog transfers promote the length of short numerics within SCREEN entry types. This behavior can be overridden by specifying EXTENDSN=NO on the catalog transfer download. The EXTENDSN= option is supported by catalog transfer of SCREEN entry types only.

Note: The V6TRANSPORT option is not needed when transferring a catalog. △

```
proc download incat=remote.scat outcat=work.scat
   extendsn=no;
run;
```

Example 9. DTS: Transferring SAS Utility Files

By using the INLIB= and OUTLIB= options with PROC UPLOAD or PROC DOWNLOAD in SAS/CONNECT, multiple SAS files may be transferred in a single step. This capability enables you to transfer an entire library or selected members of a library. You can specify which member types to transfer by using the MEMTYPE= option in one of the following statements:

- PROC UPLOAD
- PROC DOWNLOAD
- SELECT
- EXCLUDE.

If you use this option in the SELECT or the EXCLUDE statement, you can specify only one value. If you use this option in the PROC UPLOAD or the PROC DOWNLOAD statement, you can specify a list of MEMTYPE values enclosed in parenthesis.

Note: The INLIB= option must be used with the OUTLIB= option, but you can use any form (INLIB=, IN=, INDD=) of the INLIB= option with any form (OUTLIB=, OUT=, OUTDD=) of the OUTLIB= option. Also, any form (MEMTYPE=, MT=, MTYPE=) of the MEMTYPE= option may be used. △

Purpose

Previously, the only SAS files that were supported for transfer by using the method described above were SAS data sets and catalog files. This new feature also supports transfer of SQL views, MDDB files, FDB files, and DMDB files. Valid values of the MEMTYPE= option now include:

- DATA (SAS data sets)

- □ CATALOG or CAT (catalog files)
- □ VIEW (SQL views)
- □ MDDB (MDDB files)
- □ FDB (FDB files)
- □ DMDB (DMDB files)
- □ ALL (all of the above).

Programs

Example 9.1: Using the INLIB= Option in the PROC DOWNLOAD Statement

This example downloads all SAS data sets, catalog files, SQL views, MDDB files, FDB files, and DMDB files in the library WORK on the remote host and stores them in the library WORK on the local host:

```
proc download inlib=work outlib=work;
```

Example 9.2: Using the MEMTYPE= Option in the PROC UPLOAD Statement

This example uploads all MDDB and FDB files that are in the library THIS on the local host and stores them in the library THAT on the remote host:

```
proc upload inlib=this outlib=that
   memtype=(mddb fdb);
```

Example 9.3: Using the MEMTYPE= Option in the SELECT Statement

This example downloads the DMDB files TEST1 and TEST2 and the SAS data set TEST3 that are in the library WORK on the remote host and stores them in the library LOCAL on the local host:

```
proc download inlib=work outlib=local;
   select test1 test2 test3(mt=data)/memtype=dmdb;
run;
```

Example 9.4: Using the MEMTYPE= Option in the EXCLUDE Statement

This example uploads all SAS data sets, catalog files, MDDB files, FDB files, DMDB files, and SQL views except the SQL views A1, A2, A3 that are in the library LOCAL on the local host and stores them in the library REMOTE on the remote host:

```
proc upload inlib=local outlib=remote memtype=all;
   exclude a1-a3/memtype=view;
run;
```

Example 10. DTS: Distributing an .EXE File from the Remote Host to Multiple Local Hosts

Purpose

Access to remote host files by means of SAS/CONNECT makes it easy to distribute information to large numbers of local host users. Rather than distributing files on diskettes, one central file on the remote host can be copied by each local host by using SAS/CONNECT.

For example, suppose you update an executable on your PC and would like to distribute the update to other PCs in your organization. You decide that the most efficient way to update all PCs is to upload PROGRAM.EXE to the remote host, and notify each person who uses this software on their workstations that the file is available and should be downloaded. This method allows all users on the local host quick access to the updated software and eliminates passing a diskette from user to user.

Note: A SAS/CONNECT application like this one, in which an external nontext file is uploaded and then downloaded, requires the BINARY option. The BINARY option is used in the DOWNLOAD and UPLOAD procedures. The BINARY option transfers files without any character conversion (for example EBCDIC to ASCII) or insertion of record delimiters. △

Programs

Example 10.1: UPLOAD

The PROGRAM.DLL module must first be uploaded to an external file on the remote host. You start SAS/CONNECT and remote submit these statements:

```
rsubmit;
   filename rfile 'remote-host-file';
   proc upload infile='a:\program.dll'
      outfile=rfile binary;
   run;
endrsubmit;
```

This example uses a SAS FILENAME statement to identify the target file on the remote host.

Notice that the INFILE= and OUTFILE= options are used in the PROC UPLOAD statement rather than the DATA= and OUT= options. This is because the file that is being uploaded is an external file, not a SAS data set.

Execute the PROC UPLOAD program by using an RSUBMIT command. As the program executes, messages are displayed in the LOG window tracking the procedure's status. The ENDRSUBMIT command is used to terminate the remote submit.

Example 10.2: DOWNLOAD

With the PROGRAM.DLL module available on the remote host, each user on the local host at the installation can acquire the update module by downloading it from the remote host.

The process for downloading the PROGRAM.DLL module is like the process for uploading except that you invoke the DOWNLOAD procedure, and the target file is on the remote host, not the local host. For example, to copy the PROGRAM.DLL module to your directory \SAS\SASEXE, use:

```
rsubmit;
   filename rfile 'remote-host-file';
   proc download infile=rfile
      outfile='\sas\sasexe\program.dll' binary;
   run;
endrsubmit;
```

This example uses a SAS FILENAME statement to identify the target file on the remote host.

The INFILE= and OUTFILE= options are used in the PROC DOWNLOAD statement.

Execute the PROC DOWNLOAD step with the RSUBMIT command. As the file downloads, messages that track the status of the transfer are displayed in the LOG window. The ENDRSUBMIT command is used to terminate the remote submit.

Example 11. DTS: Uploading a Catalog That Contains Graphics Output

Purpose

You can use the UPLOAD and DOWNLOAD procedures to transfer catalog entries that contain graphics output. By default, the catalog entries are transferred individually and are re-created in the destination catalog in alphabetical order. However, you can alter the order or grouping of the catalog entries in the destination catalog by using SELECT statements in the UPLOAD and DOWNLOAD procedures.

Program

Assume that you have a catalog named FINANCE that has two entries, INCOME and EXPENSE, which contain graphics output. You want to download the two catalog entries that contain graphics output in the order that they are stored on the remote host. For example, you want INCOME to appear before EXPENSE, not alphabetically as the DOWNLOAD procedure would transfer them by default. In addition, you have some catalog entries that are grouped by the name GROUP1 and you want to preserve the grouping when the entries are downloaded. This program preserves the order and grouping by using SELECT statements.

```
   proc download incat=rhost.finance
      outcat=rhost.finance;
      select income expense/et=grseg;
      select group1;
   run;
```

Example 12. DTS: Downloading a Partitioned Data Set from an OS/390 Host

Purpose

This example shows users who have an OS/390 host how to download all members of a partitioned data set. Suppose you need to download a collection of SAS programs from an OS/390 host to your local host. The SAS programs are members of one partitioned data set named MFHOST.SAS.PROGRAMS. You can copy all the programs from the partitioned data set to the local host using a single DOWNLOAD procedure. An asterisk (*) can be used in the DOWNLOAD procedure as a wildcard character which greatly simplifies the code needed to transfer all members of the data set.

Program

```
❶ filename locdir
       '/unixhost/sas/programs';

❷ rsubmit;
❸    filename inpds
         'mfhost.sas.programs' shr;

❹    proc download infile=inpds('*')
          outfile=locdir('*');
❺ endrsubmit;
```

❶ The first FILENAME statement defines the fileref LOCDIR, which identifies the physical location for the files that are downloaded to the local UNIX host.

❷ The RSUBMIT statement indicates the following statement will be processed on the OS/390 host. By not specifying a *remote-session-id*, this example assumes that the OS/390 machine is your current remote host.

❸ The second FILENAME statement defines the fileref INPDS for the partitioned data set MFHOST.SAS.PROGRAMS, which contains the SAS programs that are to be downloaded to the local host.

❹ The PROC DOWNLOAD step transfers all the files in the partitioned data set on the remote OS/390 host to the library on the local UNIX host.

❺ The ENDRSUBMIT statement indicates the end of the block of statements that are submitted to the remote host for processing.

Example 13. DTS: Combining Data from Multiple Remote Sessions

Purpose

Using SAS/CONNECT to establish links to multiple remote hosts, you can access data on several hosts, draw that data together on the local host, and analyze the combined data. For example, if you have data that is stored under OS/390 in a DB2 database and related data that is stored in an ORACLE database under UNIX, you can use SAS/CONNECT in combination with SAS/ACCESS to combine that data on your local host. This example uses salary and employee data gathered from two remote hosts to illustrate the process.

Program

This example signs on to two remote hosts, downloads data from both hosts, and performs analyses on the local host. The program uses the SIGNON and RSUBMIT statements. Therefore, it can be run from a line-mode session as well as from the windowing environment.

Note: Bullets ❷ through ❺ apply to downloading both DB2 and ORACLE data. △

```
      /************************************/
      /* establish link to OS/390         */
      /************************************/
❶ options comamid=ehllapi;
   filename rlink
      '!sasroot\connect\saslink\logtso.scr';
   signon a;
      /************************************/
      /* download DB2 data using          */
      /* SAS/ACCESS view                  */
      /************************************/
❷ rsubmit a;
❸ libname db 'app.db2.views' disp=shr;
❹ proc download data=db.employee
      out=db2dat;
   run;
❺ endrsubmit;

      /************************************/
      /* establish link to UNIX           */
      /************************************/
❻ options
      remote=hrunix comamid=tcp;
         filename rlink
            '!sasroot\connect\saslink\tcpunix.scr';
         signon;

      /************************************/
      /* download ORACLE data using       */
      /* SAS/ACCESS view                  */
      /************************************/
❷ rsubmit hrunix;
❸    libname oracle '/hr/emp/records/';
❹ proc download
      data=oracle.employee out=oracdat;

   run;
❺ endrsubmit;
```

```
      /************************************/
      /* sign off both links              */
      /************************************/
❼ signoff hrunix;
   signoff a cscript=
      '!sasroot\connect\saslink\logtso.scr';

      /************************************/
      /* join data into SAS view          */
      /************************************/
❽ proc sql;
   create view joindat as
      select * from db2dat, oracdat
      where oracdat.emp=db2dat.emp;

      /************************************/
      /* create summary table             */
      /************************************/
❾ proc tabulate data=joindat
      format=dollar14.2;
      class workdept sex;
      var salary;
      table workdept*(mean sum) all,
      salary*sex;
      title1 'Worldwide Inc. Salary Analysis
              by Departments';
      title2 'Data Extracted from Corporate
              DB2 Database';
   run;

/* display graphics */
❿ proc gchart data=joindat;
      vbar workdept/type=sum
         sumvar=salary
         subgroup=sex
         ascending
         autoref
         width=6
         ctext=cyan;
      pattern1 v=s c=cyan;
      pattern2 v=s c=magenta;
      format salary dollar14.;
      title1 h=5.5pct f=duplex
          c=white
         'Worldwide Inc. Salary Analysis';
      title2 h=4.75pct f=duplex
         c=white
         'Data Extracted from Corporate DB2
          Database';
   run;
   quit;
```

❶ To sign on to a remote host, you need to provide several items of information:

- □ the remote-session id, which is specified in a REMOTE= system option or as an option in the SIGNON statement.
- □ the communications access method, which is specified by using the COMAMID= system option in an OPTIONS statement.
- □ the script file to use when signing on to the remote host. This script file is usually associated with the fileref RLINK. Using this fileref is the easiest method for accessing the script file.

When you have provided all of the necessary information, you can submit the SIGNON statement. You can specify the remote-session id in the SIGNON statement. If you omit the remote-session id from the RSUBMIT statement, the statements are submitted to the remote session that was identified most recently in a SIGNON statement, an RSUBMIT statement or command, or in a REMOTE= system option.

❷ After you have established links to two or more sessions, you can remote submit statements to any of the remote hosts by simply identifying in the RSUBMIT statement which host should process the statements. When the remote-session id has been specified by a previous statement or option, you are not required to specify the remote-session id in the REMOTE statement. This example includes the remote-session id in the RSUBMIT statements, even when the remote-session id is not required, to clarify which host is processing each group of statements.

❸ Associate a libref with the library that contains the SAS/ACCESS view of the database on the remote host.

❹ The SAS/ACCESS view can then be downloaded to the local host. Note that when you download a view of a database, a temporary SAS data set is materialized from the view and downloaded to the local host. In this example, the output data set on the local host is a temporary SAS data set.

❺ The ENDRSUBMIT statement ends the block of statements (named in the previous RSUBMIT statement) that are submitted to the remote host.

❻ To establish a second remote session, re-set the REMOTE= and COMAMID= options to values that are appropriate for the second host. You also need to reset the fileref RLINK to associate it with the script file for the second remote host.

❼ Terminate the links to both the UNIX remote host and the OS/390 remote host. Use the CSCRIPT= option to identify the script file for signing off the OS/390 host.

❽ On the local host, you can now use the SQL procedure to join into a single view the two SAS data sets that were created when you downloaded the views from the remote host.

❾ To analyze the joined data, use the name of the view on the local host in a PROC TABULATE step.

❿ If you have SAS/GRAPH on your local host, you can also use graphics procedures to analyze the view that is created from the two remote databases.

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