CHAPTER 1

Introduction

Using IT Service Vision Macros

IT Service Vision provides interactive and batch facilities with which you can access, manage, analyze, and present your performance data. The IT Service Vision batch macros provide a batch method of performing many of the same functions that you can perform within the window interface. For example, there are macros for

- starting IT Service Vision
- processing data
- reducing data
- managing your data dictionary
- creating reports.

You can write a batch script or program that contains calls to IT Service Vision macros and then submit the script as a job to be processed through the batch processing facility on your system. You can also submit and run the macros through the SAS PROGRAM EDITOR window. For more information, see the online Help topic "Using SAS Windows." The macros run outside the IT Service Vision window interface.

Within the IT Service Vision window interface, you can also perform tasks and save the macro source code for those tasks. You can then run the source code in batch mode.

The term *batch* commonly refers to any noninteractive job that is scheduled to run at a later time. That time could be a few minutes later, and therefore the job could run in parallel with your current interactive tasks. The job could also be scheduled to run overnight.

You can create a batch script by saving to a file the macro source for the commands that you select through the user interface. You can also write the batch script from scratch by using the IT Service Vision macros. Also, if you are running IT Service Vision server software on OS/390, then you can schedule a file that you made by using the IT Service Vision OS/390 Batch Utility (see "Example: The IT Service Vision OS/390 Batch Scheduler" on page 18).

Executing Macros in Batch Mode

You can submit and run IT Service Vision macros in batch mode in UNIX, OS/390, and PC environments. The term *batch* commonly refers to jobs that are run in the background, but for some environments, such as UNIX, it can also refer to a collection of jobs that are submitted to run together. In this document, batch refers to jobs that run in the background.

To run macros in batch mode, you need to create a batch file that contains calls to one or more macros. You can then submit the file and the job will be processed in batch mode on your system. You can develop and code the batch file independent of the IT Service Vision graphical user interface, or you can create the macro code by selecting tasks through the software interface and by saving the underlying macros (SOURCE) in a file. If you create the file by saving options that are selected through the window interface, then you might need to edit the batch file to include additional commands specific to your job, such as SAS, IT Service Vision, or local system statements and options. If the batch job is running on OS/390, then you should also include the appropriate JCL statements. You can then submit the batch job for background processing.

On UNIX, you can submit the job by issuing the SAS command and, within that command, by specifying the name of the file that you created, such as the following (for SAS Version 6):

```
sas /pathname/job1.sas -terminal -batch -dmsbatch -nodms -fsdevice ascii
```

In this example, *job1.sas* is the name of your batch file.

For SAS Version 8, you might use a command such as the following:

```
sas /pathname/job1.sas -noterminal
```

For OS/390, you would typically submit the batch job, which would include macros and the appropriate JCL EXEC statement to invoke SAS. For the CONFIG file, in SAS Version 6 use CPMISC(CMCONFIG) and in SAS Version 8 use CPMISC(CMCONFI8).

For both UNIX and OS/390, the batch file must include the %CPSTART macro at the beginning of the job, with MODE=BATCH specified and one or more other macros. Other commands might include, but are not limited to, a LIBNAME statement and a LINESIZE option value that is suitable for your printer.

On the PC (for example, on Windows NT), you can submit the job by issuing the SAS command and, within that command, by specifying the name of the file that you created, such as the following (for SAS Version 6):

```
c:\sas\sas.exe -dmsbatch -icon -nosplash -noxwait -sysin job1.sas
```

In this example, *job1.sas* is the name of your batch file.

For SAS Version 8, you might use a command such as the following:

```
c:\sas\sas.exe -icon -nosplash
-noxwait -noterminal -sysin job1.sas
```

This example assumes that SAS is installed in the c:\sas directory. If SAS is not installed in this location, then contact your site administrator to determine the correct location at your site.

The SAS system options that are shown or described in the previous examples are required by IT Service Vision in order to enable programs to execute without an associated window.

For a sample batch file or job for your operating environment, see the following examples:

- "Example: Creating a Batch File for UNIX" on page 16
- "Example: Creating a Batch File for PC Environments" on page 15
- "Example: Creating a Batch Job for OS/390" on page 11
- "Example: The IT Service Vision OS/390 Batch Scheduler" on page 18

If you want to run these batch programs on a regular basis, then you can schedule jobs to execute automatically by using operating environment utilities, such as `cron` on UNIX, the AT command on Windows, or a batch scheduler on OS/390.

Refer to the SAS Companion for your operating environment in order to find more information about scheduling and running batch jobs and specifying SAS system options.

Note: If you create your own SAS macros or macro variables, then avoid names that start with the letter pairs CM, CP, CV, CS, CV, or CW, unless they are specifically shown in the documentation. IT Service Vision macro names and macro variable names start with these same letter pairs, and unpredictable results might occur if you use these "reserved" macro names for your own macros and macro variables. △

For more information on using macros, see the following topics:

- For a complete list of IT Service Vision macros, refer to "IT Service Vision Macros Grouped by Task" on page 21.
- For information on what to do when batch jobs fail to execute, see "Troubleshooting Batch Jobs That Fail" on page 509. Look for information on the specific macro that you are using.
- For information on macro document conventions, see "Macro and Syntax Document Conventions" on page 3.

Macro and Syntax Document Conventions

Each macro description provides the macro name, the macro overview, the syntax (how to call the macro), information on the parameters, and any notes that relate to using the macro. When the macro name begins with the letters CP, the macro is usually available for UNIX, PC, and OS/390 operating environments. The letters CS at the front of the macro name indicate that the macro applies only to the UNIX operating environment; the letters CW indicate PC environments; and the letters CM indicate the OS/390 operating environment.

In a few instances, the macro name begins with the letters CP, but the macro is not currently available in all three operating environments. In these cases, the macro description displays the platform names in order to indicate the operating environment in which the macro is available.

When a macro is available in two or more operating environments, some of the parameters might not be available on the different hosts, might function differently on each host, or might have host-specific values. In these situations, the host-specific differences are identified by host-specific icons and are described within the parameter definition.

The "Syntax" section of each macro description lists the parameters that you can use with the macro. Macros can use the following types of parameters:

positional parameters

are identified by their position in the macro invocation. Positional parameters are not specified using a keyword; the macro understands the purpose of the parameter because the parameter occurs in a specific position in the calling sequence. In the macro syntax, positional parameters are displayed in italics. All positional parameters precede keyword parameters and should be specified in the order in which they appear in the macro syntax.

The following example macro has three positional parameters and two keyword parameters. It will be used throughout this topic:

```
%macro example(pos1, pos2, pos3, keyword1=, keyword2= );
    %put pos1=&pos1;
    %put pos1=&pos1;
    %put pos1=&pos1;
    %put keyword1=&keyword1;
    %put keyword2=%keyword2;
    %mend;
```

If you want to specify any positional parameter, then you must use a comma as a placeholder for all preceding unspecified positional parameters, as shown in the following example:

```
%example(,,value_for_pos3);
```

The next example is incorrect. If you fail to use commas as placeholders for positional parameters 1 and 2, then the value that is specified as positional parameter will be assigned to positional parameter one.

```
%example(value_for_pos3);
```

You must specify a comma after the last positional parameter if it is followed by keyword parameters. However, you do not need to specify multiple commas as placeholders for any nonspecified positional parameters that fall between the last specified positional parameter and the first keyword parameter.

In the following example, the last two commas are not required, but including them will not result in errors:

```
%example(value_for_pos1,,);
```

Likewise, in the following example, the last two commas are not required, but including them will not result in errors:

```
%example(value_for_pos1,,,keyword1=value1);
```

In the following example, commas are not required for positional parameters 2 and 3, and therefore commas are not used to take the place of those positional parameters:

```
%example(value_for_pos1,keyword1=value1);
```

However, if you do not specify any positional parameters before you specify your first keyword parameter, then you MUST NOT specify a comma. This case is shown in the following example:

```
%example(keyword1=value1);
```

The following example shows a case where you do not specify any positional parameters and where you specify the optional keyword parameters in random order:

```
%example(keyword4=value4, keyword2=value2, keyword5=value5);
```

If you do not specify any positional parameters, then no comma is required at all:

```
%example();
```

keyword parameters
: begin with a specific keyword, followed by an equal sign and then a value. You can specify these parameters in any order that you want, but they must be specified after any positional parameters. If a parameter has a default value, then this value is identified in the description of the parameter. *A comma is required to separate more than one keyword parameter, or to separate keyword parameter(s) from preceding positional parameters. However, if the macro does not have positional parameters or if you do not specify any positional parameters before your first keyword parameter, then you MUST NOT specify a comma before the first keyword parameter or the macro will fail.* If you use both positional and keyword parameters, then place a comma after the last positional parameter and between the keyword parameters:

```
%example(,value_for_pos2, keyword1=value1, keyword2=value2);
```

If you do not use positional parameters, then you MUST NOT use a preceding comma before the first keyword parameter:

```
%example(keyword1=value1, keyword2=value2);
```

Omitted keyword parameters do not need placeholder commas:

```
%example(keyword2=value2);
```

Some keyword parameters are required. Required keyword parameters are listed in alphabetical order immediately following the positional parameters.

Keyword parameters that are not required are listed in alphabetical order after the required keyword parameters in the "Syntax" section of each macro. Optional keyword parameters are enclosed in angle brackets, as shown in the syntax example below.

Suggested Typography
: Angle brackets are used to indicate optional parameters, as in <optional parameter>. Commas are displayed within the angle brackets to indicate that the comma is required only if the next parameter is specified.

For positional parameters, a comma might be required depending on whether additional positional parameters are specified following the initial positional parameter.

For keyword parameters, a comma is required preceding each keyword parameter.

In this example, assuming all parameters are optional, the syntax is documented as follows:

```
%example(  <pos1>
           ,<pos2>
           ,<pos3>
           <,optional-keyword1=value1>
           <,optional-keyword2=value2> );
```

A footnote can be included in a parameter description in order to indicate when you can omit a trailing comma after the last specified positional parameter. However, you can specify

```
%example(value_for_pos1,,,keyword1=value1);
```

instead of the simpler

```
%example(value_for_pos1,keyword1=value1);
```

and either way is correct.

Type styles and some symbols have special meanings when they are used within the macro syntax. The following list explains the style conventions that are used for syntax within this document:

UPPERCASE ROMAN	identifies macro names, keyword parameters, and values that are literals.
italics	identifies positional parameters or values that you supply.
| (vertical bar)	indicates that you can choose one value from a group. Parameters or values that are separated by bars are mutually exclusive.
... (ellipsis)	indicates that the arguments or group of arguments that follow the ellipsis can be repeated any number of times.
< > (angle brackets)	identifies optional parameters for the macro. Any parameter that is not enclosed in angle brackets is required.

Terminology

The following terms are used in macro descriptions to indicate parameter values that you supply:

physical-filename	the physical location and name of a file or library. This includes the complete directory path or high-level qualifiers and the name of the file or member. For example, on UNIX you would specify the *physical-filename* value in a format such as /sas/apps/cpe/filename; in PC environments you would specify it in a format such as c:\sas\apps\cpe\filename; and on OS/390 you would specify it in a format such as SAS.APPS.CPE(MEMBER).
pdb-name	the complete physical location of the partitioned database (PDB). This includes the complete directory path or high-level qualifiers and the PDB name. It does not include the specific library name, such as DETAIL, DAY, or WEEK. For example, the PDB= parameter in the %CPSTART macro identifies the active PDB. On UNIX you would specify the *pdb-name* value in a format such as /sas/apps/cpe/pdbname; in PC environments you would specify it in a format such as c:\sas\apps\cpe\pdbname; and on OS/390 you would specify it in a format such as SAS.APPS.CPE.PDBNAME. If the PDB name is omitted, then IT Service Vision uses the PDB that was most recently accessed.
root-location	the physical location where the PGMLIB library is installed in your operating environment. This includes the directory path or high-level qualifiers where PGMLIB is installed, but it does not

include the library name. For example, in the %CPSTART macro, you might specify the *root-location* value for the ROOT= parameter as /sas/apps/cpe on UNIX, as c:\sas\apps\cpe in PC environments, and as SAS.APPS.CPE on OS/390.

site-library	the physical location of the IT Service Vision site library in your operating environment. This includes the directory path or high-level qualifiers and the name of the site library. The site library contains system options and global information for your site. For example, in the %CPSTART macro, you might specify the *site-library* value for the SITELIB= parameter as /sas/apps/cpe/sitelib on UNIX, as c:\sas\apps\cpe\sitelib in PC environments, and as SAS.APPS.CPE.SITELIB on OS/390.

Global and Local Macro Variables

IT Service Vision provides global and local macro variables. Macro variables have built-in defaults, but you can assign a different value to a variable in order to globally control processing and simplify tasks.

Global Macro Variables

There are two types of global macro variables: automatic and static. The following global macro variables are automatic:

- CPFUTURE
- CPAUTOFT
- CPOPSYS
- CPYVAL
- CPLRMAX.

If you assign a new value to one of these automatic global macro variables during an interactive or batch session, then the new value applies only during that SAS session. The new value is not saved.

The following global macro variables are static:

- CPBEGIN
- CPEND
- CPREDLVL
- CPWHERE.

During an interactive session, if you assign a new value to one of these static global macro variables, then the new value applies during that SAS session, the value is saved (in your SASUSER library), and the value becomes the new default value for interactive and batch sessions.

During a batch session, if you assign a new value to one of these static global macro variables, then the new value applies only during that SAS session. The value is not saved.

You can change the value of a global macro variable at any time. You can also set the value to missing, as in:

```
%let CPBEGIN=;
```

To set global macro variables in a batch job, use the following format:

```
%let macro-variable = value;
```

For example:

```
%let cpwhere = 'host' ;
```

CPBEGIN

sets the initial date and time so that you can subset data by a datetime range. Specify this value by using one of the following formats: *ddmmmyy:hh:mm* or *ddmmmyyyy:hh:mm*. For example, you would specify 1:30 p.m. on September 1, 1999, as 01SEP99:13:30. You can also use the TODAY*[-nnnn unit]*@*hh:mm* format, which uses the current date value, minus *nnnn* units, at the specified hour. (Units can be day(s), week(s), month(s), or year(s). The default unit is days.) You can use this macro variable to set a begin time prior to the current date. You can also use this variable with the CPEND variable to set a datetime range.

If you do not use this macro variable, then the default value is the date and time of the earliest observation in the specified level of the specified table, as stored in the PDB's data dictionary or as obtained by scanning the data.

If you set begin and end dates in your report definition, then those dates will override any global dates. If you do not set begin and end dates in your report definition and you have global begin and end dates, then the global dates are used.

CPEND

sets the final date and time so that you can subset data by a datetime range. Use the same datetime format as used for CPBEGIN.

If you do not specify this macro variable, then the default value is the date and time of the latest observation in the specified level of the specified table, as stored in the PDB's data dictionary or as obtained by scanning the data.

If you set begin and end dates in a report definition, then those values will override any global dates. If you do not set begin and end dates in your report definition, then the global dates are used.

CPFUTURE

controls the processing of incoming data whose DATETIME variable contains a value that is more than 48 hours in the future from the current system time. (For details that are related to your specific process macro and data source, see the %CxPROCES details that follow.) The current system time is the current time on the system where data is being staged for processing into the PDB. The 48-hour buffer allows for different time zones, daylight saving time, Greenwich Mean Time, and so on.

The %CxPROCES macros use the CPFUTURE macro variable as follows:

%CMPROCES and %CPPROCES macros
: use the CPFUTURE macro variable when you process data. The default value of the CPFUTURE macro variable is used (CPFUTURE=DISCARD) unless you specify another value for CPFUTURE.

%CSPROCES and %CWPROCES macros
: use the CPFUTURE macro variable if, and only if, the COLLECTR= parameter is set to GENERIC and the

TOOLNM= parameter is set to SASDS. In this case, the MINDATE= and MAXDATE= parameters are ignored.

%CSPROCES and %CWPROCES macros
: ignore the CPFUTURE macro variable if the value for COLLECTR= is not GENERIC or if the value of TOOLNM= is not SASDS. In this case, the MINDATE= and MAXDATE= parameters on %CSPROCES or %CWPROCES are used.

Possible values for the CPFUTURE macro variable are as follows:

DISCARD	Any data with a DATETIME value 48 or more hours in the future is not staged for processing and is not processed into the PDB. A WARNING is written to the SAS log, notifying you that future data was encountered. All other data in the raw data file is staged for processing and is processed into the PDB. This value of CPFUTURE is used to prevent future data from being processed into the PDB. The future data might cause existing data to be aged out (the existing data would appear to be older than it is, in comparison with the future data). *This is the default.*
ACCEPT	All incoming data is staged for processing and processed into the PDB. If any of the data has a DATETIME value 48 or more hours in the future, then a NOTE that future data was encountered is written to the SAS log. This value of CPFUTURE is used to enable a PDB to accept future data. For example, you might want to use this setting to perform end-of-year testing with a test PDB. Note that age limits take effect from the most recent data, so dates in the future might cause at least some of the existing data to be aged out of the PDB.
TERMINATE	If any incoming data has a DATETIME value 48 or more hours in the future, then staging of the data stops, an error message is written to the SAS log, and the macro terminates. Neither the future data nor any other raw data that is in front of it or behind it in the raw data file is processed into the PDB. This value of CPFUTURE is used to prevent future data from being processed into the PDB, which might cause existing data to be aged out (the existing data would appear to be older than it is, in comparison with the future data). This value stops processing and thus calls more attention to the future data than to the DISCARD value.

If you want to use a value for CPFUTURE other than the default, then you can specify the new value with a %LET statement:

```
%let CPFUTURE=new_value;
```

In batch mode, the %LET statement must occur before the %CxPROCES macro. If you are using the window interface, then type the %LET statement in the PROGRAM EDITOR window and submit the contents of that window for processing.

To allow future data, you can set CPFUTURE=ACCEPT. This enables you to process data with “future” datetimes. Within the window interface you can submit the %LET statement from the command line or the PROGRAM EDITOR window. In batch, you would submit this statement before you submit the %CxPROCESS macro.

CPLRMAX
: keeps track of the highest value of the return code from %CPLOGRC during the current SAS session. The value of CPLRMAX is not reset by the START code in the call to %CPLOGRC. Thus, at the end of the SAS session, you can use the value of CPLRMAX as one measure of success over all the %CPLOGRC sequences in the SAS session. For more information, refer to the %CPLOGRC macro.

CPWHERE
: when reporting on data, puts data into subsets with a logical WHERE statement. Use the standard format for SAS WHERE expressions. For more information, see WHERE expressions in the *SAS Language Reference* documentation for your current release of SAS.

The following examples illustrate valid WHERE expressions:

```
%let
cpwhere=machine='host1'
%let cpwhere=machine in ('host1','host2','host3')
%let cpwhere=machine is not missing
%let cpwhere=machine is like 'host%'
%let cpwhere=machine contains 'host'
%let cpwhere=errors > 200
%let cpwhere=datetime < '01jan1999:00:00'dt
```

If you do not specify this macro variable, then a global WHERE statement is not used.

CPOPSYS
: indicates the operating environment in which you are running IT Service Vision. This macro variable is initially set by the %CPSTART macro. The default is specific to the operating environment. The one case in which you might want to change the value of this macro variable occurs when you are running on OS/390 and you want to send Web-enabled output to OS/390 directories. For more information about OS/390 directories, which are provided in OS/390 by UNIX System Services (USS) and are also known as Open Edition, see the SAS Companion for OS/390.

To set this variable, specify the following:

```
%let cpopsys=OSYS;
```

The next call to the %CPSTART macro restores the original value of CPOPSYS. To reset the value to OS/390 without using %CPSTART, specify the following:

```
%LET cpopsys=OS/390;
```

Specifying CPOPSYS=OSYS enables you to use directory names on OS/390. For example, you can specify a UNIX directory name in the HTMLDIR= and IMAGEDIR= parameters on many of the report macros when you are creating reports to display on the Web, or in the DIR= parameter on the %CMFTPSND macro.

In batch mode, the %LET statement must occur after the %CPSTART macro and before calls to any macros that generate Web-related output to OS/390 directories. In the window interface (after you invoke IT Service Vision and before you generate Web-related output to OS/390 directories), type the %LET statement in the PROGRAM EDITOR window and then submit the contents of that window for processing.

Local Macro Variables

Local macro variables are set individually for each macro when you invoke the macro. These variables are described within each macro description and syntax. For more information on a specific macro variable, refer to the appropriate macro.

You set the variables in batch mode by specifying the parameters on the macro, such as BEGIN= and END=. These variables temporarily override the values of the global macro variables, except when you specify a WHERE expression that begins with the key phrase SAME AND. In this case, the local WHERE expression is temporarily appended to the global WHERE expression. (For more information on WHERE expressions, see WHERE expressions in the *SAS Language Reference* documentation for your current release of SAS.)

Example: Creating a Batch Job for OS/390

The program in the following example performs daily processing and reduction on a PDB. At any time, you can modify a batch job by adding, deleting, or changing macros so that the program performs the tasks that you want. Many of the macro descriptions also contain examples that are specific to that macro.

Note: For similar machine-readable models of this job, see the CMJCLLD member in the PDS that is named *root-location*.CPE.CPMISC that was set up during installation at your site. For the full name of this job at your site, see your site administrator or your SAS Installation Representative. The *root-location* refers to the high-level qualifiers where the OS/390 data set is located in your operating environment. △

```
//DAILY   JOB (accounting
info),'Daily job',
//       MSGLEVEL=(1,1),TIME=20,REGION=32M,NOTIFY=
//*****************************************************************
//*                                                               *
//* This IT Service Vision for OS/390 example runs a daily job that    *
//* processes new data from SMF log file(s) and adds it to the      *
//* DETAIL level of the PDB.  It reduces data from detail level     *
//* into the DAY, WEEK, MONTH, and YEAR reduction levels.           *
//*                                                               *
//* This job now also includes a step to back up your PDB by using  *
//* IBM's DFDSS program. You can use the DFDSS DUMP command to back *
//* up volumes and data sets, and you can then use the RESTORE      *
//* command to recover them. You can also make incremental backups  *
//* of your data sets by doing a data set DUMP with RESET specified *
//* and filtering on the data-set-changed flag.                      *
//* For more information on this facility reference the IBM manual: *
//* Data Facility Data Set Services: Reference V2R5 (SC26-4389-03). *
//* You may want to update that step with the backup procedure      *
//* commonly used at your site instead.                             *
//*                                                                 *
//*=================================================================*
//*                                                                 *
//* NOTE:  This job requires that the PDB already exist.            *
//*        If you need to allocate a PDB first,                     *
//*        see the member CMPDBALC in this CPMISC pds.              *
//*                                                                 *
//* This job requires the following customization:                  *
//*                                                                 *
//*  1) Change the JOB card to a valid job card for your system.    *
//*                                                                 *
//*  2) Do a global change of YOUR.SMF.DATA to the data set         *
//*     name of your SMF data sets.                                 *
//*                                                                 *
//*  3) Do a global change of MXG.USERID.SOURCLIB to the name       *
//*     of your customized MXG source library.                      *
//*                                                                 *
//*  4) Do a global change of MXG.MXG.SOURCLIB to the name of the   *
//*     PDS containing the original MXG source library.             *
//*                                                                 *
//*  5) Do a global change of MXG.MXG.FORMATS to the name           *
//*     of your MXG format library.                                 *
//*                                                                 *
//*  6) Do a global change of YOUR.SASCPE to the high-level         *
//*     qualifiers of this application on your OS/390 system.      *
//*     Make sure this qualifier has been set correctly for the     *
//*     CONFIG parameter.                                           *
//*                                                                 *
//*  7) Do a global change of YOUR.CPE.PDB to the high-level        *
//*     qualifier for your PDB. This PDB should have one or more    *
//*     tables already in it.                                       *
//*                                                                 *
//* Plus,in the BACKUP step using IBM's ADRDSSU program:            *
//*                                                                 *
//*  8) Modify characteristics on the DD named TAPE allocation      *
//*     and/or on the DUMP command according to your needs or       *
//*     site-specific rules. For instance, you may want to create   *
//*     and use GDGs for the backup tape data sets.                 *
//*                                                                 *
//*  9) ADRDSSU may not work when any of the PDB's SAS data         *
//*     libraries span multiple volumes.  For more information,     *
//*     check with SAS Technical Support.                           *
//*                                                                 *
//* Plus, in the LOAD step:                                         *
//*                                                                 *
//* 10) For SAS Version 6, use CPMISC(CMCONFIG).  For SAS           *
//*     Version 8, use CPMISC(CMCONFI8).                            *
//*                                                                 *
//*******************************************************************
```

```
//BACKUP  EXEC PGM=ADRDSSU
//TAPE     DD DSN=your.cpe.pdb,DISP=(OLD,KEEP),
//            UNIT=cart,RETPD=30
//SYSPRINT DD SYSOUT=A
//SYSIN   DD *
  DUMP DATASET(INCLUDE(your.cpe.pdb.*)) OUTDD(TAPE) -
       SHARE COMPRESS
//*
//LOAD   EXEC SAS,WORK='10500,2000',
//       CONFIG='your.sascpe.CPMISC(CMCONFIG)'
//SMF     DD DSN=your.smf.data,DISP=SHR
//SYSIN   DD DATA,DLM='$$'

*-------------------------------------------------------------------*
  * This application will not run unless %CPSTART has been executed *
  * %CPSTART allocates this application and the PDB.                *
  * The MXGLIB= and MXGSRC= parameters are required for the         *
  * MXG software tool used  within %CMPROCES below.                 *
  * The ROOTSERV= and SHARE= parameters should only be used if      *
  * SAS/SHARE is being used with this application.                  *
  * If the _RC= parameter is nonzero from %CPSTART,                 *
  * check the explanatory message in the SAS log.                   *
  *-------------------------------------------------------------------* ;


  /* verify root= pdb= mxglib= mxgsrc= in the following macro */
  %CPSTART(MODE=BATCH,
           ROOT='your.sascpe.',
           ROOTSERV=,
           PDB='your.cpe.pdb.',
           DISP=OLD,
           SHARE=N/A ,
           MXGLIB=mxg.mxg.formats,
           MXGSRC=('mxg.userid.sourclib' 'mxg.mxg.sourclib'),
           _RC=cpstrc
          );

  %PUT CPSTART return code is &cpstrc;


  *--------------------------------------------------------------*
  * The %CMPROCES macro processes data into the PDB that was   *
  * established by the %CPSTART macro above.                   *
  * As no table list is specified as the second parameter all  *
  * the tables in the PDB for this collector with a table kept *
  * indicator of YES will be processed.                        *
  * TOOLNM=MXG and COLLECTR=SMF parameters must be specified.  *
  * If the _RC= parameter is nonzero from %CMPROCES,           *
  * check the explanatory message in the SAS log.              *
  *--------------------------------------------------------------* ;


  %CMPROCES(,
            COLLECTR=SMF,
          TOOLNM=MXG,
          _RC=cmprrc
         );

%PUT CMPROCES return code is &cmprrc;



*------------------------------------------------------------*
* The %CPREDUCE macro reduces data in the current PDB detail *
* level into the day, week, month and year reduction levels. *
* The variables and statistics kept are defined in the data  *
* dictionary.                                                *
* As no table list is specified as the first parameter,      *
* reduction will be performed on all the tables in the PDB.  *
* If the _RC= parameter is nonzero from %CPREDUCE            *
* check the explanatory message in the SAS log.              *
*------------------------------------------------------------* ;


%CPREDUCE(,_RC=cpredrc);

%PUT CPREDUCE return code is &cpredrc;

$$
//
```

Processing Notes

1 Before you submit the job for the first time, you must perform the following steps, typically from the graphical user interface within the application:
   a Estimate the space requirements for your PDB and allocate space accordingly.
   b Add table definitions to the PDB, as required. If the table definitions that are explicitly specified by a table list in the %CMPROCES macro are not in the PDB, then the %CMPROCES macro automatically adds those table definitions to the PDB by copying them from the master data dictionary.
   c Edit the table definitions to adjust durations for retention of DETAIL, DAY, WEEK, MONTH, and YEAR data, as required.
   d Add, update, or delete variables in the tables, as required.
   e Edit the variable statistics to adjust which statistics are requested for DAY, WEEK, MONTH, and YEAR levels, as required.
   f Verify that all tables and variables that you want to use are marked KEPT=YES.

2 The %CPSTART() macro invokes IT Service Vision and allocates the PDB.

   The MODE= parameter must be set to *batch*.

   The ROOT= parameter specifies the high-level qualifiers in the data set names for this application. For example, if IT Service Vision is located in a SAS library that is installed at your site with the name *your.sascpe.pgmlib*, then use *your.sascpe* as the value for the ROOT= parameter. Similarly, the PDB= parameter specifies the high-level qualifiers in the OS/390 data set names for the SAS data libraries in the PDB. For example, if the PDB is in libraries that are all named *your.sascpe.pdbname*, where *pdbname* varies depending on the library name (such as DETAIL, DAY, WEEK, MONTH, and YEAR), then use *your.sascpe* as the value for the PDB= parameter.

The MXGLIB= parameter specifies the fully qualified name of the SAS library that contains the SAS formats that are provided by MXG. The MXGSRC= parameter specifies fully qualified names of one or more PDSs that contain MXG SOURCLIB source entries and user overrides to these source entries. List the PDSs in search order. If the same member name exists in more than one of the PDSs, then the member that is used is the first one that is encountered. Thus, typically, list the PDS that has your source entries in front of the PDS with the MXG source entries.

3 If the batch job contains the %CMPROCES macro, then you must specify the name of the OS/390 data set that contains the logged SMF-style data in one of the following ways:

- with a DD statement
- with a FILENAME statement, such as

  ```
  filename smf SMF_style_log_name;
  ```

  If you use the FILENAME statement, then place it before the %CMPROCES macro invocation.
- as the first positional parameter in the %CMPROCES macro.

4 The %CMPROCES macro reads logged SMF data from the log file and processes it into the detail level of the PDB for the specified tables. Specifying a list of table names is optional. If no names are specified, then the macro processes all tables in your PDB with KEPT=YES.

The COLLECTR= parameter in the %CMPROCES macro can be set to SMF or GENERIC.

The PDB into which the data is processed is the PDB that is specified in the %CPSTART macro by the PDB= parameter. This is referred to as the active PDB.

The TOOLNM= parameter in the %CMPROCES macro can be set to MXG or SASDS. This application uses the SAS library and the PDSs that are specified in the %CPSTART macro by the MXGLIB= and MXGSRC= parameters.

5 The %CPREDUCE macro reduces the data in the DETAIL level of the current PDB into the DAY, WEEK, MONTH, and YEAR reduction levels. Specifying a table name is optional. If no table name is specified, then the macro reduces all tables in your PDB that have KEPT=YES and AGELIMIT > 0 in at least one of the DAY, WEEK, MONTH, or YEAR levels. The *level* must also have at least one variable with a selected statistic and KEPT=YES.

Example: Creating a Batch File for PC Environments

The .bat file in the following example starts SAS in batch mode from the DOS command line. The comments within the code describe how to use the script. Many of the macro descriptions also contain examples that are specific to that macro.

For more information on setting up your daily process and reduction job(s) or your data collector, see the online IT Service Vision Setup documentation.

```
@echo off
Rem - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Rem Doc:    Start SAS in batch with proper options for IT Service Vision
Rem Usage:  sasbat infile
Rem         Where "infile" is the simple (LE 8 characters, nopathname)
```

```
Rem         first name of the files read and written by SAS:
Rem               .sas is appended to "infile" for the input program
Rem               .log is appended to "infile" for the SAS log
Rem               .alt is appended to "infile" for the alt SAS log
Rem Example:  cd \mypgms
Rem           sasbat mysaspgm
Rem  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
set INFILE=%1%

Rem  Prefix SAS command with "start /w" on Windows 95 so script will wait
Rem On Windows NT, "start /w" is not necessary
Rem The two lines below are for SAS Version 6.  For SAS Version 8,
Rem remove -dmsbatch and add -noterminal
start /w C:\sas\sas -sysin %INFILE% -altlog %INFILE%.alt -awstitle "ITSV
Batch" -dmsbatch -icon -nosplash -noxwait

Rem Note common exit codes
for %%c in (0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16) do if errorlevel %%c set
EXITCODE=%%c

Rem Did SAS terminate gracefully?
find "NOTE: SAS Institute Inc., SAS Campus Drive" %INFILE%.alt > nul
if not errorlevel 1 goto NORMTERM
echo SAS was interrupted before completing termination processing.
:NORMTERM

Rem Complain appropriately based on the exit code
if not %EXITCODE%==0 goto ERRORS

Rem Scan the SAS log since we might still have errors with zero exit
Rem code
find "ERROR: " %INFILE%.alt > nul
if not errorlevel 1 goto ERRORS
echo SAS terminated without errors (EXITCODE eq 0).
goto EXIT

:ERRORS
echo SAS terminated with errors (EXITCODE ge %EXITCODE%).
goto EXIT

:EXIT
@echo on
```

Example: Creating a Batch File for UNIX

The UNIX shell script in the following example starts SAS in batch mode from the UNIX command line and displays a message that is based on the return code from the SAS session. The comments within the code describe how to use the script. Many of the macro descriptions also contain examples that are specific to that macro.

For more information on setting up your daily process and reduction jobs or your data collector, see the online IT Service Vision Setup documentation.

```
#!/bin/ksh
# ---------------------------------------------------------------+
# Copyright (C) 1996 by SAS Institute Inc., Cary, NC 27512-8000
# Name:     sasbat
# Doc:      Run SAS with proper options for ITSV batch mode.
# ---------------------------------------------------------------+

PROGNAME='/bin/basename $0'

# -----------------------------------------------------------------------
# First, be helpful
# -----------------------------------------------------------------------
if test "$#" = "0"
   then echo " "
        echo "Usage: $PROGNAME [sasfile]"
        echo " "
        echo "Where 'sasfile' is the simple (no dots, no dirname)"
        echo "first name of the files read and written by SAS:"
        echo " "
        echo "  .sas is appended to 'sasfile' for the input program"
        echo "  .log is appended to 'sasfile' for the output SAS log"
        echo "  .lst is appended to 'sasfile' for SAS printed output"
        echo " "
        echo "Example: cd /u/myid/mypgms"
        echo "         sasbat mysasprog"
        echo " "
        echo "This reads the SAS program in /u/myid/mypgms/mysasprog.sas,"
        echo "creates a SAS log in /u/myid/mypgms/mysasprog.log,"
        echo "and writes printed output in /u/myid/mypgms/mysasprog.lst."
        exit 0
fi


# -----------------------------------------------------------------------
# Begin
# -----------------------------------------------------------------------
echo $PROGNAME: Starting at 'date'

INFILE=$1

rm $INFILE.log 2> /dev/null
rm $INFILE.lst 2> /dev/null

# -----------------------------------------------------------------------
# Start SAS
# Note:  The line below is for SAS Version 6.
#        For SAS Version 8, use
#             sas  -noterminal $INFILE.sas
# -----------------------------------------------------------------------
sas    -terminal -batch -dmsbatch -fsdevice ascii.vt100  $INFILE.sas

# -----------------------------------------------------------------------
# Advertise exit code
# -----------------------------------------------------------------------
```

```
EXITCODE=$?

# Did SAS terminate gracefully?
grep "NOTE: SAS Institute Inc., SAS Campus Drive" $INFILE.log > /dev/null
if test $? -ne 0
   then echo SAS was interrupted before completing termination processing.
fi

# Complain appropriately based on the exit code
if test $EXITCODE -eq 0
   then # We can still have errors even with zero exit code
        grep "Errors printed on page" $INFILE.log > /dev/null
        if test $? -eq 0
           then echo "SAS terminated with errors (though EXITCODE eq 0)."
           else echo "SAS terminated without errors (EXITCODE eq 0)."
        fi
   else echo "SAS terminated with errors (EXITCODE eq $EXITCODE)."
fi

# ---------------------------------------------------------------------------
# That is all
# ---------------------------------------------------------------------------
echo $PROGNAME: Ending at ‘date‘

exit $EXITCODE
```

Example: The IT Service Vision OS/390 Batch Scheduler

If you use IT Service Vision with OS/390, then you can use the Batch Utility, which is available from the server interface on OS/390, in order to save and schedule jobs to be processed. You can then submit the job to run in batch mode by creating a file that calls the %CPBATCH macro. For example, if you run the job on a Tuesday, then the Batch Utility runs all the tasks with a schedule of Daily, all the tasks with a schedule of Weekday, and all the tasks with a schedule of Tuesday. The order in which it runs the tasks is determined by the priorities that are set in the Batch Utility. For more information about the OS/390 batch scheduling subsystem, refer to the Batch Help Index that is available from the server interface on OS/390.

The following job submits the tasks that were set up by using the Batch Utility. The comments within the program describe areas of the code that you should change in order to customize the code.

Note: For a similar machine-readable model of this job, see the CMJCLSCD member in the PDS that is named *root-location*.CPE.CPMISC that was set up during installation at your site. For the full name of this PDS at your site, see your site administrator or your SAS Installation Representative. The *root-location* refers to the high-level qualifiers where IT Service Vision is installed at your site. △

```
//SCHEDULE JOB (accounting info),'schedule job',
//       MSGLEVEL=(1,1),TIME=20,REGION=32M,NOTIFY=
//*******************************************************************
//*                                                                 *
//* This IT Service Vision example runs a daily OS/390 job.         *
//* It runs the list of tasks that have previously been             *
//* set up by your site administrator.                              *
//* To set up the schedule use the Batch Utility                    *
//* within the IT Service Vision application.                        *
//* Schedule the job to run after the nightly SMF dump.             *
//*                                                                  *
//*================================================================*
//*                                                                  *
//* NOTE:  This job requires that the PDB already exist.            *
//*        If you need to allocate a PDB first,                      *
//*        see the member CMPDBALC in this CPMISC pds.               *
//*                                                                  *
//* This job requires the following customization:                  *
//*                                                                  *
//*  1) Change the JOB card to a valid job card for your system.    *
//*                                                                  *
//*  2) Do a global change of YOUR.SMF.DATA to the data set         *
//*     name of your SMF data sets.                                  *
//*                                                                  *
//*  3) Do a global change of MXG.USERID.SOURCLIB to the name       *
//*     of your customized MXG source library.                       *
//*                                                                  *
//*  4) Do a global change of MXG.MXG.SOURCLIB to the name of the   *
//*     PDS containing the original MXG source library.              *
//*                                                                  *
//*  5) Do a global change of MXG.MXG.FORMATS to the name           *
//*     of your MXG format library.                                  *
//*                                                                  *
//*  6) Do a global change of YOUR.SASCPE to the high-level         *
//*     qualifiers for this application on your OS/390 system.      *
//*     Make sure this qualifier has been set correctly for the     *
//*     CONFIG parameter AND for the SITELIB data library,          *
//*     which contains your site specific settings.                 *
//*                                                                  *
//*  7) Do a change of YOUR.CPE.PDB to the high-level               *
//*     qualifier for your PDB. This PDB should have one or more    *
//*     tables already in it if your site administrator             *
//*     has included inquire task(s) for the batch utility.         *
//*                                                                  *
//*******************************************************************
//SASCPE EXEC SAS,WORK='10500,2000',
//       CONFIG='your.sascpe.CPMISC(CMCONFIG)'
//SMF     DD DSN=your.smf.data,DISP=SHR
//SYSIN   DD DATA,DLM='$$'

  *----------------------------------------------------------*
  * To run this application you must first submit %CPSTART.    *
  * %CPSTART allocates this software and the PDB.              *
  * The MXGLIB= and MXGSRC= parameters are required for the    *
  * MXG software tool used  within %CMPROCES below.            *
  * The SITELIB= parameter must be specified.                  *
  * The SITEACC= parameter must be set to OLD disposition.     *
  * The ROOTSERV=, SHARE=, and SITESHR= parameters should only *
  * be used if SAS/SHARE is being used at                      *
  * your site.                                                 *
  * If the _RC= parameter is nonzero from %CPSTART,            *
```

```
   * check the explanatory message in the SAS log.              *
   *------------------------------------------------------------* ;


   /* verify root= pdb= sitelib= mxglib= mxgsrc= in the following */
   %CPSTART(MODE=BATCH,
            ROOT='your.sascpe.',
            ROOTSERV=,
            PDB='your.cpe.pdb.',
            DISP=OLD,
            SHARE=N/A,
            SITELIB='your.sascpe.SITELIB',
            SITEACC=OLD,
            SITESHR=,
            MXGLIB=mxg.mxg.formats,
            MXGSRC=('mxg.userid.sourclib' 'mxg.mxg.sourclib'),
            _RC=cpstrc
           );

   %PUT CPSTART return code is &cpstrc;


   *------------------------------------------------------------*
   * The following %CPBATCH macro runs the list of scheduled    *
   * tasks that have been set up by your site administrator.    *
   * You must code the ENV= parameter and capitalize its value. *
   *------------------------------------------------------------* ;
   %CPBATCH(ENV=BACK);

 $$
 //
```

Processing Notes

1 Before you submit the job for the first time, you must perform the following steps, typically from the menu interface of this application:

- **a** Estimate the space requirements for your PDB and allocate space accordingly. You can accomplish this through the client user interface.
- **b** Add table definitions to the PDB, as required.
- **c** Edit the table definitions to adjust durations for retention of detail-, day-, week-, month-, and year-level data, as required.
- **d** Add, update, or delete variables in the tables, as required.
- **e** Edit the variable statistics to adjust which statistics are requested for day, week, month, and year levels, as required.
- **f** Verify that all tables and variables that you want to use are marked KEPT=YES.

2 The %CPSTART macro invokes the IT Service Vision software and allocates the PDB.

The MODE= parameter must be set to *batch*.

The ROOT= parameter specifies the high-level qualifiers in the data set names for IT Service Vision software. For example, if the PGMLIB library is located in a SAS library that is installed at your site with the name *your.root-location.pgmlib*, then use *your.root-location* as the value for the ROOT= parameter. Similarly, the PDB= parameter specifies the high-level qualifiers in the data set names for the SAS data libraries in the PDB. For example, if the PDB is in libraries that are all named *your.root-location.pdbname*, where *pdbname* varies depending on the library name (such as DETAIL, DAY, WEEK, MONTH, and YEAR), then use *your.root-location* as the value for the PDB= parameter.

The MXGLIB= parameter specifies the fully qualified name of the SAS library that contains the SAS formats that are provided by MXG. The MXGSRC= parameter specifies fully qualified names of one or more PDSs that contain MXG SOURCLIB source entries and user overrides to these original entries. List the PDSs in search order. If the same member name exists in more than one of the PDSs, then the member that is used is the first one that is encountered. Thus, typically, list the PDS that has your modified source entries in front of the PDS with the MXG source code.

3 The %CPBATCH macro runs the tasks that were set up by using the **`Batch Utility`** item on the main menu of the interactive interface to run on days that match the day of this run.

IT Service Vision Macros Grouped by Task

IT Service Vision provides macros that can be run in batch mode to perform many of the same tasks that can be performed through the user interface. For example, you can use the batch macros to process your data, manage and update your PDB, create report definitions on your data, and generate reports on your data.

For information on using any of the macros during recovery from an error condition, see "Troubleshooting Batch Jobs That Fail" on page 509.

The following list of macros and overviews is divided into groups that are based on the types of tasks that the macros perform:

- "Macros for Building and Managing the PDB" on page 28
- "Macros That Are Used for Analyzing Data" on page 201
- "Using the %CPDDUTL Macro" on page 429 and "%CPDDUTL Macro Control Statements" on page 433.

The Reporting macros can run on either your client system or your server system. However, you can run the other macros only on your server system, because many of the macros require write access to the PDB.

For more information on IT Service Vision macros, see "Using IT Service Vision Macros" on page 1 and "Macro and Syntax Document Conventions" on page 3.

CHAPTER
2

Administration Macros