ABSTRACT
This paper discusses changes in the SAS® System interface to DB2® between Version 5 and Version 6, with an emphasis on converting applications. The SAS/ACCESS® interface to DB2 with Version 6 includes new features, such as the ACCESS and DBLOAD procedures and the DB2 interface engine to manipulate data in DB2 tables. In Version 5, SAS/DB2® software used procedures such as DB2EXT, DB2UTIL, and DB2LOAD to access DB2. This paper is intended for users who have written applications for SAS/DB2 in Version 5 and are beginning to work with the SAS/ACCESS Interface to DB2 in Version 6.

INTRODUCTION
The SAS/ACCESS interface to DB2 brings many new features and concepts to the way SAS programs interface with DB2. The implementation of the interface to DB2 is governed by a concept that is introduced in Version 6 called Multiple Engine Architecture. This architecture allows SAS procedures and DATA steps to access data in DB2 tables directly and in a transparent manner. It will no longer be necessary to create an extracted copy of a DB2 table to use in your SAS applications. The SAS/ACCESS interface to DB2 also makes use of the increased flexibility and efficiency that the SAS WHERE statement provides in specifying selection criteria.

Users of the Version 5 SAS/DB2 product will need to learn these concepts to take advantage of the new features. The procedures DB2EXT, DB2UTIL, and DB2LOAD that composed the interface to DB2 in Version 5 are not available in Release 6.06. Applications that ran under Version 5 need to be converted to run under the Release 6.06 architecture. Some applications will be easy and straightforward to convert; other applications will be more difficult. This paper covers strategies that can be used to convert existing Version 5 applications.

Although more compatibility with Version 5 is planned for the next release after 6.06, some applications may benefit enough from the new Version 6 features to warrant immediate conversion. Other applications might be better off waiting for the increased compatibility of the next release. This paper also gives guidelines on which applications might fall into these categories.

OVERVIEW OF MULTIPLE ENGINE ARCHITECTURE
With Version 6 of the SAS System, the concept of Multiple Engine Architecture (MEA) is introduced. Engines are a set of routines that are used to read and write to a file in a particular format.

Native Engines vs. Interface Engines
Two categories of engines defined by Multiple Engine Architecture are native engines and interface engines.

Native Engines
are routines to read, write, update, and maintain files specific to the SAS System. Native engines are the routines that are used to do the internal I/O to SAS data libraries.

Interface Engines
manipulate files that are specific to another vendor's software. The DB2 engine is an example of an interface engine.

Engines are behind the scenes of the SAS application. The application in Release 6.06 can now access data in many formats directly without having to be concerned about the underlying format. To access data in a DB2 table, a special kind of view is set up. This view is a virtual SAS data set that does not contain the actual data values but instead contains a definition on how to get the data values when needed. Then the application just references the view, and the appropriate engine (in this case the DB2 interface engine) is selected automatically and is used to retrieve the data and present it to the application for processing.

The ACCESS Procedure and Descriptor Files
The ACCESS procedure is a full-screen procedure used to define the special views that are going to be used by your application. The first step is to set up a descriptor of the DB2 table. This descriptor is called an access descriptor and is a SAS file with a member type of access that resides in a SAS data library. This access descriptor contains the information about one DB2 table or DB2 view that PROC ACCESS requires to create the views that will be used in the application. The information in the access descriptor includes the authorization ID, the table name, and the columns in the table and their data types. The information also enables you to specify your own default values for SAS variables and formats. You can also delete columns you do not want to be available for the views that will be created later.

Once the access descriptor has been created, you can then create the views of the table that will be used in the application. These views are called SAS/ACCESS view descriptors. View descriptors are also created with PROC ACCESS and are SAS files with a member type of view that resides in a SAS data library. They contain the information that the DB2 engine needs to access the data in the DB2 table for this specific view. This information includes the authorization ID and table name, the actual columns that will be retrieved, and the SAS variable names and formats that will be used. When defining a view descriptor, you can also specify a WHERE clause for a specific selection criterion or an ORDER BY clause to retrieve the rows in a specific order.

Using View Descriptors in Applications
Creating these descriptors is basically a one-time operation. The view descriptor that you have defined fits into the definition of a SAS data set. This view descriptor, when used with the appropriate engine, can present the image of a SAS data set to the application. The ability to present the image of a SAS data set allows the view descriptor to be used in the SAS application in the same way any SAS data set would be used as input for a SAS procedure or DATA step. For example, suppose an access descriptor has been created for the SASDEMO.EMPLOYEES table. Then suppose a view descriptor is created from the access descriptor, given the name EMPLOYEE, and created in the SAS data library referenced by the libref PERSONNL. A report on the number of employees per department could be generated with the following SAS statements:

PROC PRINT DATA=PERSONNL.EMPLOYEE;
  TABLES DEPT;
  RUN;
A listing of the employees in the Shipping Department could be generated with these SAS statements:

```sas
PROC PRINT DATA=PERSONNEL.EMPLOYEE;
WHERE DEPT LIKE 'SHP';
```

Here the SAS WHERE statement is used to subset the observations to only those employees in the Shipping Department. The DB2 engine uses the selection criteria to only retrieve only the specified rows from the DB2 table.

### Updating and Creating DB2 Tables

Data in a DB2 table can be updated through view descriptors with procedures such as the FSEDIT, FSVIEW, SQL, and APPEND procedures. To create a new DB2 table the DBLOAD procedure is used. This procedure is very similar to PROC DB2LOAD in Version 5.


### STRAIGHTFORWARD CONVERSIONS

The main focus of your conversion from Version 5 to Version 6 will center around taking an application that used a SAS/DB2 procedure and an actual SAS data set and changing it to an application that uses a view descriptor and accesses the data in the DB2 table directly. Let’s first deal with the more straightforward conversions.

#### Single Table Extractions

The most simple SAS/DB2 application in Version 5 extracts data from one DB2 table using PROC DB2EXT and uses the extracted data set to do analysis and create a report from the data. To create the proper descriptors to run this application under Version 6, follow the steps outlined below:

1. **Create an access descriptor:** This is done with PROC ACCESS and is based on the same DB2 table as PROC DB2EXT in the Version 5 application. At this point, you may want to protect certain columns so they cannot be selected when creating subsequent view descriptors. You may also want to specify your own defaults for SAS variable names and formats.

2. **Create the view descriptor:** This view descriptor is based on the access descriptor just created and is also created by using PROC ACCESS. The view descriptor should be defined with the same SAS variable names and formats that the Version 5 data set has. You will even want to store the view descriptor in the Version 6 library with the same name that the Version 5 data set has. This view descriptor essentially becomes the replacement for the Version 5 data set.

This process of creating the descriptors is only done once. Also note that you can create more than one view descriptor based on the same access descriptor. If you have another application that uses a different subset of the same DB2 table, you will only need to create a view descriptor of the new subset based on the existing access descriptor.

Now you have a view of the DB2 table that mirrors the data set that PROC DB2EXT would extract in the Version 5 application. There are three changes that need to be made to the application itself:

1. Remove the PROC DB2EXT step from the application.
2. Remove the allocation to the Version 5 SAS data library where the extracted data set would have been stored.
3. Add the allocation to the Version 6 SAS data library where the view descriptor has been stored. Note that this library should be allocated with the same libref as the Version 5 library.

When the application runs, it will access the data directly from the DB2 table.

There is another approach you might want to take if your application extracts the data and then manipulates them before doing the analysis. Most likely you will not want to manipulate the data in the table directly; instead you will just want to manipulate your copy. One feature of PROC ACCESS allows you to extract the data described by a view into an actual SAS data file. Currently, this is the only feature of PROC ACCESS that can be executed in a batch environment. To convert this type of application, start the same as outlined above by creating an access and a view descriptor that mirror the SAS data set used in the Version 5 application. The only difference is that this time you will give the view descriptor a unique name in the Version 6 SAS data library. In the application, replace the PROC DB2EXT step with a PROC ACCESS step that specifies the name of the view descriptor in the VIEWDESC= option and uses the same OUT= option that the PROC DB2EXT step used. For example, if the code in your Version 5 application is:

```sas
PROC DB2EXT lib=DATA.SAS;,
SELECT • FROM AUTHIO.APPLICATIO~.
```

the steps to convert it are as follows:

1. Create an access descriptor based on the DB2 table AUTHID.APPLICATION..TABLE.
2. Create a view descriptor that defines the SAS variables and formats the same as the data set DATA.SAS.EXTDATA. This view descriptor is stored in the Version 6 library referenced by the libref DSG0LIB and with the member name of EXTVIEW.
3. Remove the PROC DB2EXT step and replace it with a PROC ACCESS step like this:

```sas
PROC ACCESS VIEWDESC=DSG0LIB.EXTVIEW OUT=DATA.SAS.EXTDATA;
```

You have now made a copy of your data that your application can manipulate independently of the DB2 table.

#### The DBLOAD Procedure

Another application that is straightforward to convert uses PROC DBLOAD to load a SAS data set into a DB2 table. PROC DBLOAD in Version 6 is very similar in function and syntax to PROC DB2LOAD.

Most of the differences in syntax between PROC DB2LOAD and PROC DBLOAD are in the PROC statement itself. Most of the PROC DB2LOAD statement options become separate statements in PROC DBLOAD. Examples of these options are the TABLE=, SSID=, COMMIT=, LIMIT=, and LOAD options. Some options that are not supported are the mapping data set, SQL file, and error file options.

Most of the DB2LOAD procedure statements are the same in PROC DBLOAD. Examples of these statements are the TYFE, NULLS, IN, and SQL statements. One change is that the REPLACE statement is now the RENAME statement.
Converting an existing PROC DB2LOAD job is a matter of changing the syntax. For example, if your existing application is:

PROC DBLOAD DATA=DATALIB.LOADDATA TABLE=AUTHID.NEW_TABLE
SSID=DB2P COMMIT=500;
LOAD;

The equivalent PROC DBLOAD statements are:

PROC DBLOAD DATA=DATALIB.LOADDATA DBMS=DB2;
SSID=DB2P;
COMMIT=500;
TYPE=SASVAR1=INTEGER SAVAR2=DECIMAL(10,2);
REPLACE SAVAR2=DB2COL2 SAVAR4=DB2COL4;
NULLS SAVAR3=N;
LOAD;

The FSEDIT Procedure

Applications that use PROC FSEDIT are straightforward to convert as well. PROC FSEDIT in Version 6 works with a view descriptor in the same way it works with a SAS data file. The steps to create the access and view descriptors are similar to the steps taken when creating descriptors for a single table extraction. First, an access descriptor is created that defines the DB2 table. Second, a view descriptor is created that has the SAS variable names and formats that you want displayed on the PROC FSEDIT screen. The only change that needs to be made in the application is to change the PROC FSEDIT statement so that the TABLE = option that referred to the DB2 table is now a DATA= option that refers to the view descriptor in the Version 6 SAS data library. For example, if your existing PROC FSEDIT statement is:

PROC FSEDIT TAlA"DESCLIB.EMP_VIEW;

the code for your Version 6 application would look like this:

PROC FSEDIT DATA=DESCLIB.EMP_VIEW;

Here the libref DESCLIB refers to a Version 6 SAS data library that holds the view descriptor EMP_VIEW. By taking the defaults for SAS variable names and formats when creating the view descriptor, your PROC FSEDIT display will be similar to the Version 5 display of a DB2 table. You can also change the variable names and formats when creating the view descriptor if desired.

The INSERT Function of the DB2UTIL Procedure

An application that uses the INSERT function of PROC DB2UTIL is also straightforward to convert. The APPEND procedure is a base SAS procedure that is used to add observations to an existing SAS data set. In Version 6, PROC APPEND also adds observations to an existing DB2 table through a view descriptor. Once again, your first step is to use PROC ACCESS to create an access and a view descriptor on the table that will receive the new rows. Use the MAPTO statement from the DB2UTIL step as a guide when defining the names of SAS variables in the view descriptor. In the application, remove the PROC DB2UTIL step and replace it with a PROC APPEND step that references the view descriptor in the BASE= option. For example, if your Version 5 code is:

PROC DB2UTIL DATA=DATALIB.INSDATA TABLE=AUTHID.NEW_TABLE
MAPTO=SASVAR1=DB2COL1 SASVAR2=DB2COL2
UPDATE;

you would do the following steps to convert this job:

1. Create an access descriptor based on the DB2 table AUTHID.TABLE..NAME.
2. Create a view descriptor that defines the DB2 table with the same variable names as the SAS data set. Using the MAPTO statement above as a guide for the DB2 column named DB2COL1, give it the SAS variable name SASVAR1, and for the DB2 column DB2COL2 give it the SAS variable name SASVAR2. This view descriptor would be stored in the Version 6 data library allocated with the libref DESCLIB and with the member name of NEWVIEW.

3. Remove the PROC DB2UTIL step from the application and replace it with a PROC APPEND step such as the following:

PROC APPEND BASE=DESCLIB.NEWVIEW DATA=DATALIB.INSDATA;

Another method to convert this application is to use PROC SQL instead of PROC APPEND to add the new rows to the table. It is still necessary to create the access and view descriptors as described above. The difference is that the PROC DB2UTIL step in the existing application is replaced with a PROC SQL step that uses an INSERT statement. This PROC SQL step is as follows:

PROC SQL;
INSERT INTO DESCLIB.NEWVIEW (SASVAR1, SASVAR2)
SELECT SASVAR1, SASVAR2 FROM DATALIB.INSDATA;

For additional information on using the INSERT statement of PROC SQL, see the SAS Guide to the SQL Procedure, Usage and Reference, Version 6, First Edition.

MORE DIFFICULT CONVERSIONS

Some applications that are fairly common to Version 5 SAS/DB2 users are not as straightforward to convert. You might want to consider not converting these applications immediately. If you want to use some of the new features in Version 6 for areas other than SAS/ACCESS interface to DB2, you may be able to divide your application between Version 5 and Version 6.

Extraction From a Multi-Table Join

One common application that is more difficult to convert is an application that uses PROC DB2EXT to extract data from a multi-table join. The difficulty arises from the fact that PROC ACCESS creates access and view descriptors that reference only one DB2 object (that is, a DB2 table or a DB2 view). However, in Version 6, the SQL procedure is able to create a SAS data view that is a join of multiple view descriptors. One way to convert your application is to create access and view descriptors for each DB2 table involved in the join, and then use PROC SQL to create the SAS data view that will be used by your application. For example, if your PROC DB2EXT step is:

PROC DB2EXT OUT=DATALIB.EXSDATA
SELECT * FROM TAB1 INNER JOIN TAB2 ON COL1 = COL2;

the steps to convert the application with this strategy would be as follows:

1. Create an access descriptor based on the DB2 table TAB1.
2. Create a view descriptor that defines the DB2 table TAB1 with the SAS variable names desired for this application. This view is stored in the Version 6 data library referenced by the libref DESCLIB with a member name of TAB1VIEW.
3. Create an access and a view descriptor for the DB2 table TAB2 in the same manner as they were created for TAB1. This view descriptor is stored with a member name of TAB2VIEW.
4. Use the following PROC SQL statements to create a SAS data view that joins the two view descriptors:

385
The application can then use the SAS data view to read the data; it can manipulate it, then the SAS data view need not be created. In this case you would replace the PROC DB2EXT step with a PROC SOL step that uses a CREATE TABLE statement instead of a CREATE VIEW statement, as in this example:

```
PROC SOL;
CREATE TABLE DATALIB.EXTDATA AS
SELECT * FROM DESCLIB.TAB1, DESCLIB.TAB2
WHERE COL1 = COL2;
```

Note that in PROC SQL statements the term table is referring to an actual SAS data file, not a DB2 table.

Another method to convert this type of application is to create a DB2 view that does the join. PROC ACCESS can create an access and subsequent view descriptor based on a DB2 view. The process for creating descriptors based on a DB2 view is exactly the same as creating descriptors for a DB2 table. Once the DB2 view has been defined, the process for converting this application then becomes the same as converting for an application that does a single table extraction. One advantage of this technique is possible better performance because DB2 does the join processing. One disadvantage might be increased work by the DBA staff to create and maintain the DB2 view.

The UPDATE and DELETE Functions of the DB2UTIL Procedure

Applications that use the UPDATE and DELETE functions of PROC DB2UTIL are difficult to convert as well. There are two procedures that update a DB2 table in a non-full-screen mode. PROC SOL performs updates and deletions on a DB2 table through a view descriptor that has been defined for the table. With PROC DBLOAD you can submit SQL statements directly to DB2 that do updates or deletions on the DB2 table.

You can convert this type of application by using some method to generate the SQL UPDATE or DELETE statements that PROC DB2UTIL would have generated. To get an idea of what the SQL statements look like, just add the SQLOUT = option to the PROC DB2UTIL statement of your current Version 5 application. This causes PROC DB2UTIL to write a copy of the SQL UPDATE or DELETE statements that it is generating to an external file.

Once you know what the format of the SQL statements are, you can write a DATA step that generates these same SQL statements using the values from your data set. Once the statements are in the file, they can be included with the %INCLUDE statement into a PROC DBLOAD step with a %INCLUDE statement. The DATA step and PROC DBLOAD statements that do this are as follows:

```
FILENAME TEMPSOL 'TEMPFILE';
DATA _null_
FILENAME TEMPSOL;
PUT 'SOL UPDATE AUTHID.TABLE1 SET DB2COL1 = ' SASVAR1 '
WHERE DB2COL1 = ' SASVAR2 ';;
RUN;
%INCLUDE TEMPSOL;
RUN;
```

You could also use a macro to generate these statements. If you create an access and a view descriptor for the table, you could use PROC SOL to do the updates or deletions instead of PROC DBLOAD.

MAPPING DATA SETS

Some of your Version 5 SAS/DB2 applications may make use of mapping data sets. Support for these mapping data sets is not implemented in Release 6.06. The primary purpose of the mapping data sets in Version 5 was to allow you to build the specifications for the extract or load using the full-screen panels and then to do the actual processing in batch. Most of this functionality is present in Version 6 through the use of access and view descriptors. PROC ACCESS is a full-screen procedure where you specify the table, columns, formats, variable names, and selection criteria that will be in your descriptor. Once the descriptors are built, they can be used in any environment.

USING VERSION 5 AND VERSION 6 TOGETHER

In some cases you may decide not to convert an application immediately. If other portions of your application need to run under Version 6, one possibility is to use Version 5 and Version 6 together. To do this, divide your application into separate job steps. One job step will execute Version 5 to use the SAS/DB2 interface, and the second job step will execute Version 6 to run the rest of your application. Note that Version 6 has an engine that allows access to SAS data libraries in Version 5 format. Having your application split into two separate job steps will incur more overhead because of the need to initialize the SAS System twice, but this may not outweigh the effort required to fully convert the application.
CONCLUSION

Multiple Engine Architecture in Version 6 offers SAS/ACCESS users much in the way of new features and flexibility. To take advantage of this new architecture, SAS/DB2 users must convert Version 5 applications to the Version 6 syntax. In some cases this conversion is straightforward; in other cases it is more difficult. This paper has tried to give guidelines for deciding which applications should be converted first and strategies for doing the conversion.

REFERENCES

SAS/ACCESS Interface to DB2: Usage and Reference, Version 6, First Edition

Emily P. Wallace’s “Database Interfaces under the Version 6 Engine Architecture” in Proceedings of the Fourteenth Annual SAS Users Group International Conference
Jake Jacobs’ “Writing Efficient SAS/DB2 Applications in Version 6 of the SAS System” in SUGI 15 Proceedings
Howard Plemmons’ “Performance Considerations for the Database Engines” in SUGI 15 Proceedings

SAS/DB2 is a trademark and SAS/ACCESS is a registered trademark of SAS Institute Inc., Cary, NC.
DB2 is a trademark of IBM Corporation.