

**I** Viewing your files is much easier now with the availability of new FRAME objects in SAS/AF® software such as the Catalog Entry Viewer and the External File Viewer. However, is it possible to query the name of the file currently being viewed, and is it also possible to print the file?

**O** You can query the filename and print the file currently displayed. In order to write your own methods to implement these capabilities, you must first subclass the model associated with your particular viewer and add your own methods.

This article demonstrates how to add your own methods for printing and demonstrates one possible way to define these methods. You can implement the methods differently, providing more error checking along with other parameters that you may need to carry out the task at your site. Basic knowledge of subclassing and writing your own methods is assumed.

In these examples, the subclassed entries and methods are stored in the SASUSER.CLASSES catalog.

### Subclassing the Catalog Entry Model Class

Issue the command `BUILD SASUSER.CLASSES` to edit the catalog. Edit `CATENTRY.SCL` and specify `SASHELP.FSP.CATENTDE` as the Parent class. Select `METHODS` under `Additional Attributes` to open the `METHODS` window. Select `Actions` and `Add mode on` to add the following new methods:

NAME	Source Entry	Run Label in SCL Entry
GET-ENTRY	SASUSER.CLASSES.CATENTRY.SCL	GETENTRY
PRINT	SASUSER.CLASSES.CATENTRY.SCL	PRINT

Select `Actions` then `Add mode off` when you've completed adding your methods. You should see your new methods listed preceded by a + symbol indicating that these are new methods.

Now you must define your methods. Edit `CATENTRY.SCL` and enter the following code and then compile the SCL entry. Because these are your own methods, you can alter the code or add any

parameters you may need. `_ENAME_` refers to the entry name associated with the catalog entry viewer excluding the entry type. `_ETYPE_` contains the entry type.

```
length entryname $ 27 entrytype $ 8;
GETENTRY:
  method entry $ 35;
    entryname=getni temc(-sel f-, '-ename-');
    entrytype=getni temc(-sel f-, '-etype-');
    entry=entryname||'.'||entrytype;
  endmethod;

PRINT:
  method formname $ 35 dest $ 8 optional= fileref $ 8 ;
    call send(-sel f-, 'get-entry', entry);
    if upcase(dest)='PRINTER' then do;
      rc=preview(' copy', entry);
      rc=preview(' print', formname);
      rc=preview(' clear' );
    end;
    if upcase(dest)='FILE' then do;
      if fileref(fileref)<=0 then do;
        rc=preview(' copy', entry);
        rc=preview(' print', formname, fileref);
        rc=preview(' clear' );
      end;
    end;
  endmethod;
```

The `GET_ENTRY` method returns the four-level name of the catalog entry currently being displayed.

The `PRINT` method requires two parameters and accepts an optional third parameter. The first parameter specifies the name of the form to use when printing the catalog entry. By default, if no form is specified, the `DEFAULT` form is used. Specify `''` for `FORMNAME` if you want to use the default form. The second parameter is the destination. If `DEST='PRINTER'`, then the file is sent to your printer specified in your form. If `DEST='FILE'`, then the method determines whether the file reference is currently defined. The `PRINT` method copies the contents from the catalog entry into the `PREVIEW` buffer and prints the contents using



the PREVIEW function. The PRINT method also executes the GET\_ENTRY method to determine the name of the entry.

### Attaching Your Model to the Catalog Entry Viewer Object

In order to invoke your methods, you must first attach your model to the viewer. The FRAME used in this example contains a Catalog Entry Viewer called OBJ1 and a push button called PRINTCAT. The FRAME views an OUTPUT entry called WORK.TEST.A.OUTPUT. In the SCL code for your FRAME application, include the following statements:

```
INIT:
❶ catview=instance(loadclass
    ('sasuser.classes.catentry.class'));
    call notify('obj1', '_attach_', catview);
❷ call send(catview, '_set-entry-', 'work.test.a.output');
    return;

PRINTCAT:
❸ rc=filename('out', 'flat.fil');
    call send(catview, 'print', '', 'file', 'out');
    rc=filename('out', '');
    return;
```

- ❶ The LOADCLASS and INSTANCE functions load the CATENTRY class and create an instance of it. The instance is stored in the CATVIEW variable. The \_ATTACH\_ method attaches this instance of the CATENTRY model class to the Catalog Entry Viewer.
- ❷ The SEND routine invokes the \_SET\_ENTRY\_ method and specifies the name of the entry to display in the Catalog Entry Viewer.
- ❸ The FRAME contains a push button called PRINTCAT that when selected executes the FILENAME function to assign a fileref to point to an external file. The SEND routine invokes the new PRINT method and specifies the FILE as the destination and OUT as the fileref. If no form is specified, DEFAULT is used. Clear the fileref upon completion.

You can also execute the GET\_ENTRY method from within the SCL code associated with the FRAME entry as follows:

```
length entry $ 35;
call send(catview, 'get-entry', entry);
put entry=;
```

### Subclassing the External File Model Class

Issue the command BUILD SASUSER.CLASSES to edit the catalog. Edit FILEVIEW.CLASS and specify SASHELP.FSP.EXFILEDE as the Parent class. Select METHODS under Additional Attributes to open the METHODS window. Select Actions and Add mode on to add the following new methods:

NAME	Source Entry	Run Label in SCL Entry
GET-FILE	SASUSER.CLASSES. FILEVIEW.SCL	GETFILE
PRINT	SASUSER.CLASSES. FILEVIEW.SCL	PRINT

Select Actions then Add mode off when you've completed adding your methods. You should see your new methods listed preceded by a + symbol indicating these are new methods.

Now you must define your methods. Edit FILEVIEW.SCL, enter the following code and then compile the SCL entry. Because these are your own methods, you can alter the code or add any parameters you may need. \_FNAME\_ contains the physical filename or the fileref of the external file associated with the External File Viewer. \_FTYPE\_ contains an 'N' if the FILENAME is a physical filename or an 'F' if it contains a fileref. If FILENAME contains a fileref, use the PATHNAME function to return the physical filename.

```
length ftype $ 1;
GETFILE:
    method filename $ 80;
        ftype=getntemp(-self-, '_ftype-');
        filename=getntemp(-self-, '_fname-');
        if ftype='F' then filename=pathname(filename);
    endmethod;

PRINT:
    method optional = hostcmd $ 35 ;
        call send(-self-, 'get-file', filename);
        if hostcmd='' then
            rc=system('print' || filename);
        else
            rc=system(hostcmd || ' ' || filename);
    endmethod;
```

*continued on next page*



The GET\_FILE method returns the name of the external file or the fileref of the file currently being displayed.

The PRINT method has one optional parameter. This parameter can specify the host command that is executed along with any options necessary to print the file. In this example, the default host command is called PRINT and will execute if no parameters are specified. You can alter the method to include any parameters or to specify the specific host command used to send files to the printer at your site. The PRINT method executes the GET\_FILE method to determine the name of the entry.

## Attaching Your Model to the External File Viewer Object

In order to invoke your methods, you must first attach your model to the viewer.

The FRAME used in this example contains an External File Viewer called OBJ1. The FRAME views an external file called EXTFILE.SAS. In the SCL code for your FRAME application, include the following:

INIT:

```
❶ fileview=instance(loadclass
                    ('sasuser.classes.fileview.class'));
   call notify('obj1', '_attach-', fileview);
❷ call send(fileview, '_set-file-', 'extfile.sas', 'n',
            'NOCC');
❸ call send(fileview, 'print');
return;
```

- ❶ The LOADCLASS and INSTANCE functions load the FILEVIEW class and create an instance of it. The instance is stored in the FILEVIEW variable. The \_ATTACH\_ method attaches this instance of the FILEVIEW model class to the External File Viewer.
- ❷ The SEND routine invokes the \_SET\_ENTRY\_ method and specifies the name of the external file to display in the External File Viewer.
- ❸ The SEND routine invokes the new PRINT method. Because no parameter is specified, it uses the default host command of PRINT to send the file to the printer.

You can also execute the GET\_FILE method from within the SCL code associated with the FRAME entry as follows:

```
length filename $ 50;
call send(fileview, 'get-file', filename);
put filename=;
```

Adding methods to your subclassed model and attaching your model to the viewer is simple. For more information on the models, viewers, and subclassing, refer to *SAS/AF Software: FRAME Applications Development Concepts, Version 6, First Edition*. For additional training consider attending the course *Object-Oriented Programming Using SAS/AF Software*. Copyright © 1996 by SAS Institute Inc.

*Answer provided by Yvonne Selby. Yvonne is a senior technical support analyst at SAS Institute where she specializes in SAS/AF and SAS/EIS® software and the FRAME entry.*

SAS/AF and SAS/EIS are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.