Overview

The SAS Catalog class defines an object-oriented interface to SAS catalogs. A SAS Catalog object corresponds to a SAS catalog and contains information about the catalog, including its name, its associated SAS library, a set of catalog attributes, and a list of its catalog members. The list of catalog members is a list of lists; each item in the list of members of a SAS Catalog object describes a SAS catalog entry.

Parent:
  sashelp.fsp.sasfile.class

Class:
  sashelp.fsp.Catalog.class

Using the Catalog Class

You can create a SAS Catalog object this way: when you instantiate a SAS Catalog object, only the object identifier is created. You then add information about a SAS catalog to the object. This step is referred to as initializing the object. To initialize a SAS Catalog object, use the _setup method. This example creates a SAS Catalog object, CATOBJ, and initializes it to SASUSER.PROFILE:

```sas
    catclass=loadclass('sashelp.fsp.catalog.
           class');
    catobj=instance(catclass);
    call send (catobj, '_setup_', 'sasuser.
           profile');
```

Note: Unless an example includes LOADCLASS, INSTANCE, and _setup, the example assumes that you have instantiated a SAS Catalog object and initialized it using these statements. ▲

If you list the contents of CATOBJ, you see

```sas
  (_class=2651
   DESC='SAS Catalog Class'
```
LIBRARY and ATTRIBUTES are list identifiers. Notice that at this point, they have a value of 0, an invalid list identifier. The reason is that these values are not initialized by _setup. Use _getLibrary and _getAttributes to initialize these lists.

### Methods

Methods specific to the SAS Catalog class are described here. The SAS Catalog class inherits methods from the SAS Generic File and Object classes. All methods from the SAS Generic File class are overridden. Object methods are described in the Object class.

### Dictionary

#### _copy

Copies a SAS catalog

---

**Syntax**

CALL SEND (catalog-object-id, '_copy', target-catalog-name<, new-catalog-object>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target-catalog-name</td>
<td>C</td>
<td>specifies the two-level name of the target catalog to which the catalog that is associated with catalog-object-id is to be copied</td>
</tr>
<tr>
<td>new-catalog-object</td>
<td>N</td>
<td>returns the object identifier of the new SAS Catalog object created for target-catalog-name</td>
</tr>
</tbody>
</table>

**Details**

The _copy method copies the SAS catalog that is associated with catalog-object-id to a new SAS catalog. If the second parameter is specified, a new SAS Catalog object, associated with target-catalog-name, is created and its object identifier is returned.

**Example**

This example copies the SAS catalog, SASUSER.CAT1, to SASUSER.CAT2, creates a new SAS Catalog object for SASUSER.CAT2, and returns that object identifier in CATOBJ2:
```sas
catclass=loadclass('sashelp.fsp.catalog.
class');
catobj1=instance(catclass);
catobj2=0;
call send (catobj1, '_setup_', 'sasuser.
cat1');
call send (catobj1, '_copy_', 'sasuser.cat2',
catobj2);
```

### _copyMember

Copies a catalog entry contained by a SAS Catalog object to another entry in the same SAS Catalog object.

#### Syntax

```sas
CALL SEND (catalog-object-id, '_copyMember', source-member-name,
target-member-name<, new-object-id>);
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>target-member-name</td>
<td>C</td>
<td>specifies the one-level name (entryname) of the catalog entry to which the source is to be copied</td>
</tr>
<tr>
<td>new-object-id</td>
<td>N</td>
<td>returns the object identifier for the new SAS Catalog Entry object that is associated with target-member-name</td>
</tr>
</tbody>
</table>

#### Example

This example copies SASUSER.CAT1.MYENTRY.PROGRAM to SASUSER.CAT1.OTHER.PROGRAM:

```sas
catclass=loadclass('sashelp.fsp.catalog.
class');
catobj=instance(catclass);
call send (catobj, '_setup_', 'sasuser.cat1');
call send (catobj, '_copy_', 'sasuser.cat2',
catobj2);
```

### See Also

- _copy

### _delete

Deletes the SAS catalog associated with a SAS Catalog object and clears the SAS Catalog object.
**Syntax**

CALL SEND (catalog-object-id, '_delete');

**Details**

_delete clears the association with the SAS catalog and deletes the SAS catalog. However, it does not delete the SAS Catalog object itself.

**See Also**

_deleteMember

---

**_deleteMember**

Deletes a catalog entry from the catalog associated with a SAS Catalog object

**Syntax**

CALL SEND (catalog-object-id, '_deleteMember', entry-name);

**Argument Type Description**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry-name</td>
<td>C</td>
<td>specifies the two-level name (entryname.entrytype) of the catalog entry to be deleted</td>
</tr>
</tbody>
</table>

**Example**

This example deletes SASUSER.CAT1.MYENTRY.PROGRAM:

```plaintext
catclass=loadclass('sashelp.fsp.catalog.
                         class');
catobj=instance(catclass);
call send (catobj, '_setup_', 'sasuser.cat1');
call send (catobj, '_delete_member_',
                      'myentry.program');
```

**See Also**

_delete

---

**_getAttributes**

Returns a list of attributes for the catalog associated with a SAS Catalog object
**Syntax**

CALL SEND (catalog-object-id, '_getAttributes', attributes-list);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>attributes-list</td>
<td>N</td>
<td>returns the identifier of an SCL list that contains the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>items in Table 88.1 on page 1651. Each item in the list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>corresponds to a SAS catalog attribute.</td>
</tr>
</tbody>
</table>

**Table 88.1 _getAttributes List Items**

<table>
<thead>
<tr>
<th>List item</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCESS</td>
<td>C</td>
<td>the user access level to the catalog, either 'WRITE' or 'READ'</td>
</tr>
<tr>
<td>ENTRIES</td>
<td>N</td>
<td>the number of catalog entries</td>
</tr>
</tbody>
</table>

**Example**

This example creates a SAS Catalog object, CATOBJ, and initializes it to SASUSER.PROFILE. The _getAttributes method returns the attributes for CATOBJ.

```sas
catclass=loadclass('sashelp.fsp.catalog.class');
catobj=instance(catclass);
call send(catobj, '_setup_', 'sasuser.profile');
attrlist=makelist();
call send(catobj, '_getAttributes', attrlist);
```

If you use PUTLIST to list the contents of ATTRLIST, you see something similar to

```
( ACCESS='WRITE'
  ENTRIES=31
)[2965]
```

In this example, SASUSER.PROFILE contains 31 entries, and the user has write access.

**_getCatentry**

Creates a SAS Catalog Entry object for a given catalog entry and returns its object identifier.
Syntax

CALL SEND (catalog-object-id, '_getCatentry', new-catentry-name, catentry-object);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catentry-object</td>
<td>N</td>
<td>returns the object identifier for a catalog member contained by catalog-object-id</td>
</tr>
</tbody>
</table>

Details

The _getCatentry method creates and initializes a SAS Catalog Entry object corresponding to a SAS catalog entry contained in catalog-object-id, and it returns the object identifier of the new SAS Catalog Entry object.

Example

This example creates a SAS Catalog Entry object for SASUSER.CAT1.MYENTRY.PROGRAM and returns its object identifier in CATENTRY:

```sas
    call send (catobj, '_get_catentry_',
               'myentry.program', catentry);
```

_getFullname

Returns the two-level, fully-qualified name for the catalog that corresponds to a SAS Catalog object

Syntax

CALL SEND (catalog-object-id, '_getFullname', catalog-fullname);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog-fullname</td>
<td>C</td>
<td>returns the fully-qualified name (libref:catalog) for the catalog that is associated with catalog-object-id</td>
</tr>
</tbody>
</table>

Example

This example returns "SASUSER.CAT1" in fullname:

```sas
    length fullname $35;
    call send (catobj, '_get_fullname_',
               fullname);
See Also

_getName

__getLibrary

Creates and returns a SAS Library object identifier that corresponds to the SAS library that contains the catalog for this SAS Catalog object

Syntax

CALL SEND (catalog-object-id, '_getLibrary', library-object-id);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>library-object-id</td>
<td>N</td>
<td>returns the object identifier for a SAS Library object that corresponds to the SAS library that contains the catalog for catalog-object-id</td>
</tr>
</tbody>
</table>

Details

The SAS Library object that is created, library-object-id, can be accessed and manipulated by all of the methods defined in the SAS Library class.

Example

This example returns the object identifier for the SAS Library object that contains the catalog associated with CATOBJ in LIBRARY:

```
call send (catobj, '_get_library_', library);
```

__getMembers

Conditionally retrieves a list of member names contained in a SAS Catalog object

Syntax

CALL SEND (catalog-object-id, '_getMembers', members-list<<, select-exclude>>, item-mask<<(, 'APPEND')>>);
_getMembers and _loadMembers perform the same function except that _getMembers enables you to select the members that are loaded into the members-list.

By default, _getMembers clears all items from members-list and fills it with the SAS Catalog object members. If ‘APPEND’ is specified, the SAS Catalog object members are
appended to the existing members-list. The conditions defined by the select-exclude, item-mask, and options arguments determine the items that are returned in the list.

Examples

Example 1: Select-Exclude Lists  The following call to _getMembers returns all catalog entries from the SAS Catalog object, CATALOG, except those of type PROGRAM:

```sas
call send (catobj, '_get_members_', entrylist, "objtype='^program'");
```

The following call to _getMembers returns all catalog entries from the SAS Catalog object, CATALOG, with a type of FRAME or SCL:

```sas
call send (catobj, '_get_members_', entrylist, "objtype='frame scl'");
```

Example 2: Item Masks  The following call to _getMembers returns all catalog entries from the SAS Catalog object, CATALOG. Each entry in entrylist has the following item names: OBJ NAME, OBJ DESC, MODIFIED.

```sas
call send (catobj, '_get_members_', entrylist, "", "objname objdesc modified");
```

The following call to _getMembers returns all catalog entries from the SAS Catalog object, CATALOG. Each entry in ENTRYLIST has all item names except MODIFIED.

```sas
call send (catobj, '_get_members_', entrylist, "", "^modified");
```

Example 3: Creating an object for a catalog  This example creates an object for the SAS catalog, SASUSER.CAT1, and retrieves information for all entries in SASUSER.CAT1 except those of type PROGRAM.

Each entry in ENTRYLIST contains the LIBNAME, CATNAME, OBJ NAME, OBJ TYPE, and OBJ DESC items.

```sas
entrylist=makelist();
call send (catobj, '_get_members_', entrylist, "objtype='^program'", 'libname catname objname objtype objdesc');
```

If you list the contents of ENTRYLIST, you see

```sas
( LIBNAME='SASUSER' CATNAME='PROFILE' OBJNAME='PASSIST' OBJTYPE='SLIST' OBJDESC='User profile' ){2693}
( ){2661}
```

_getMembers returns the LIBNAME, CATNAME, OBJ NAME, OBJ TYPE, and OBJ DESC for each entry in SASUSER.PROFILE whose entry type is not PROGRAM. In this example, SASUSER.PROFILE contains only one entry, PASSIST.SLIST. Your SASUSER.PROFILE may contain many other entries.
See Also

_loadMembers

__getName

Retrieves the name of the catalog that corresponds to the SAS Catalog object

**Syntax**

CALL SEND (catalog-object-id, '_getName', catalogname);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalogname</td>
<td>C</td>
<td>returns the one-level catalog name that is associated with catalog-object-id</td>
</tr>
</tbody>
</table>

**Example**

This example returns the one-level catalog name for SASUSER.PROFILE (PROFILE) into CATALOGNAME:

```plaintext
length catalogname $8;
call send (catobj, '_get_name_', catalogname);
```

See Also

__getFullname

__loadMembers

Unconditionally loads the information about the members of a SAS Catalog object into an SCL list

**Syntax**

CALL SEND (catalog-object-id, '_loadMembers', members-list<, 'APPEND'>);
Argument | Type | Description
---|---|---
members-list | N | specifies the identifier of an SCL list. Each entry in members-list corresponds to a SAS catalog member and is a sublist containing the items in Table 2. Specify members-list as a numeric variable that is initialized either to 0 or to a valid list identifier. If the value of members-list is 0, then _getMembers creates a new list, fills it with the members of the SAS Catalog object, and returns the list identifier. If members-list is nonzero, then it must be valid list identifier; otherwise, the program halts with an error.

‘APPEND’ | C | appends members to members-list

Details
_loadMembers and _getMembers perform the same function except that _loadMembers loads all members from a SAS catalog into members-list. The _getMembers method enables you to select the members that are loaded.

The form of members-list is the same as the form specified in _getMembers. See _getMembers for this class.

By default, _loadMembers clears all items from members-list and fills it with the SAS Catalog object members. If ‘APPEND’ is specified, then the SAS Catalog object members are appended to the existing members-list.

Example

This example retrieves the information for all catalog entries in SASUSER.CAT1 and then appends the information for all catalog entries in SASUSER.CAT2 to ENTRYLIST:

```sas
   catclass=loadclass('sashelp.fsp.catalog.
                        class');
   catobj1=instance(catclass);
   catobj2=instance(catclass);
   entrylist=makelist();
   call send (catobj1, '_setup_', 'sasuser.
                     cat1');
   call send (catobj2, '_setup_', 'sasuser.
                     cat2');
   call send (catobj1, '_load_members_',
              entrylist);
   call send (catobj2, '_load_members_',
              entrylist, 'append');
```

See Also

_getMembers

rename

Renames the SAS catalog that is associated with a SAS Catalog object
Syntax

CALL SEND (catalog-object-id, '_rename', new-catalog-name);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new-catalog-name</td>
<td>C</td>
<td>specifies the new one-level catalog name for the catalog associated with catalog-object-id</td>
</tr>
</tbody>
</table>

Example

This example renames the SAS catalog SASUSER.CAT1 to SASUSER.CAT2:

```plaintext
catclass=loadclass('sashelp.fsp.catalog.
class');
catobj=instance(catclass);
call send (catobj, '_setup_', 'sasuser.
cat1');
call send (catobj, '_rename_', 'cat2');
```

See Also

(renameMember)

(renameMember)

Renames a catalog entry in a SAS catalog object

Syntax

CALL SEND (catalog-object-id, '_renameMember', old-member-name,
new-member-name<, member-description>);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>new-member-name</td>
<td>C</td>
<td>specifies the one-level name (entryname) to be used in renaming old-member-name</td>
</tr>
<tr>
<td>member-description</td>
<td>C</td>
<td>specifies the new description for the member. The description can be up to 40 characters.</td>
</tr>
</tbody>
</table>

Example

This example renames SASUSER.CAT1.ENTRY.PROGRAM to SASUSER.CAT1.OTHER.PROGRAM:

```plaintext
catclass=loadclass('sashelp.fsp.catalog.
class');
```
catobj=instance(catclass);
call send (catobj, '_setup_','
    sasuser.cat1');
call send (catobj, '_rename_member_','
    entry.program', 'other');

See Also
rename

_setup

Initializes a SAS Catalog object with information about a SAS catalog

Syntax
CALL SEND (catalog-object-id, '_setup', catalog-name);

<table>
<thead>
<tr>
<th>Argument</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>catalog-name</td>
<td>C</td>
<td>specifies the two-level name of an existing SAS catalog to link to</td>
</tr>
</tbody>
</table>

Details

For an example of using _setup, see “Using the Catalog Class” on page 1647.