CHAPTER

32

Examples of Indirect Messaging

Introduction

This chapter outlines the general steps for implementing a sample distributed application by using indirect messaging.

Initialize the DOMAIN Server for Collection Services

Message queues belong to collections, such that QueueA in Collection1 is distinct from QueueA in Collection2. The collection manager is responsible for managing the distinction between the queues and for starting the queue manager for processing individual messages. An example of PROC DOMAIN follows:

```
libname domain ".";
proc domain collection id=/shr9;
run;
```

After starting, PROC DOMAIN continues to run until PROC ADMIN is used to terminate it.

Loading a Station Instance

A client application must set up a *station instance.* This station instance is used to communicate with a specific collection at the DOMAIN server.

```
stationid = loadclass('sashelp.connect.station');
stationInst = instance(stationid);
```

Open a Station Instance

Open the station instance with the appropriate collection name at the DOMAIN server.

```
call send(stationInst, '_open', 'collectionName', rc);
```

Note: The *collectionName* parameter identifies either a new collection that will be created dynamically by the DOMAIN server or a collection that already exists in the DOMAIN server. The collection name is uniquely associated with a queue. You can have multiple queues with the same name, as long as they exist in different collections. △

Open a Queue

An application generally wants to access one or more queues. In order to open a queue, you must supply your station instance, the queue's name, the open mode, and one or more queue attributes. The station instance has already been instantiated, and the queue name is whatever you use to refer to the queue. Queues can have up to 32 byte names.

The open mode is determined by the way the queue is used. There are four options for open mode:

- FETCH (retrieve messages),
- FETCHX (retrieve messages exclusively),
- BROWSE (read messages),
- DELIVERY (send messages).

You may specify only one mode when you open a queue. If you need to open the queue in multiple modes, you will need to open it multiple times. For example, if you want the queue for DELIVERY and FETCH, you must open it two times.

The open attributes are used to specify characteristics of the queue and are applied when the queue is opened.

```
queueID = loadclass('sashelp.connect.queue');

   /*************************************/
   /* Open MyQueue for DELIVERY.        */
   /*************************************/
deliveryQ = instance(queueID);
call send(deliveryQ, '_open', stationInst,
   "MyQueue", "DELIVERY", rc, "DYNTEMP");

   /*************************************/
   /* Open MyQueue for FETCH.           */
   /*************************************/
fetchQ = instance(queueID);
call send(fetchQ, '_open', stationInst,
          "MyQueue", "FETCH", rc, "POLL");
```

Note: By default, if you query for a message and there is no message, you must wait until a message is received. If you do not want to wait, use the POLL attribute to open the queue. △

Send a Message to a Queue

After a queue is opened for delivery, the application can send message(s) to that queue. The parameters are the various pieces of data and the descriptions that compose the message. Optionally, a message type can be sent to indicate the characteristic of the message, such as the number and type of parameters. However, the MSGTYPE parameter would need to be choreographed between the sending and receiving applications to work properly.

Header information for the message can contain user-defined information in addition to a time stamp that shows when the message was removed from the queue. A list of the attachments to accompany the message can also be sent. If there are no attachments, a 0 should be specified. The last parameters are pieces of data that comprise the message. There can be 0 or more numeric or character parameters that are sent.

```
msgtype = 42;
header = makelist();
rc = setnitemc(header, "Add to inventory.",
               "DESCRIPTOR");
attachlist = 0;
product = "widgets";
quantity = 35;
action = "ADD";

call send(deliveryQ, '_send', msgtype, header,
          attachlist, rc, product, quantity,
          action);
```

Query a Queue

The application can check the status of an open queue. If a message is on the queue, the event type from the query is set to DELIVERY, the type of message is returned, as well as header information and a list of attachments.

```
call send(fetchQ, '_query', eventtype,
          msgtype, header, attachlist, rc);
```

Receive a Message from a Queue

If there is a message on the queue, the event type from the query would be set to DELIVERY. The client can now get the message header and the message content from the queue. However, the sender and receiver must choreograph the use of the message type (*msgtype*) so that the correct number and type of parameters are received.

```
if (eventtype = "DELIVERY") then do;

   if (msgtype = 42) then do;
      code="";
      num =0;
```

```
      action="";

      call send(fetchQ, '_recv', rc,
                code, num, action);
   end;
end;

else if (eventtype = "NO_MESSAGE") then do;
end;
```

Close a Queue

When the application has completed its task, the open queue(s) should be closed. By default, permanent queues are not deleted.

```
call send(deliveryQ, '_close', rc);
call send(fetchQ, '_close', rc);
```

Close Station Instance

The application no longer needs services from this DOMAIN server collection.

```
call send(stationInst, '_close', rc);
```

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

ISBN 1–58025–477–2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999