CHAPTER

38

The DOMAIN Server

Introduction

The DOMAIN server technology delivers protocol-independent messaging and eliminates the requirement for licensing, configuring, and supporting multiple protocol stacks in a given environment.

The DOMAIN server manages intercommunications in SAS/CONNECT by providing four services:

Collection Management
: separates queues into collections and is responsible for starting the queue manager for processing individual messages.

Queue management
: allocates the queues, maintains access information for each queue, and administers the messages that belong to each queue.

Agent scheduling
: is a client/server-based implementation of a periodic job scheduler and can also provide on-demand execution.

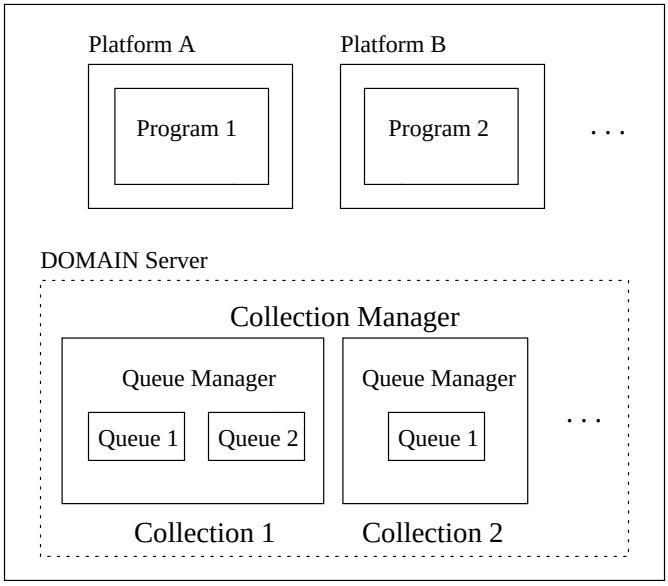
Protocol gateway
: allows message exchange between SAS sessions that run in a network environment and use different network access methods. A network is made up of one or more logical domains. A logical domain maps a topological area according to the communications protocol or access method that it supports.

Collection Manager

The DOMAIN server is used to initialize the collection manager by using PROC DOMAIN with the COLLECTION option. The collection manager manages the grouping of queues into collections in addition to starting the queue manager for processing individual messages.

The collection manager must be initialized before applications can use the indirect-messaging facility. The following figure illustrates the basic structure of the SAS indirect-messaging facility. In this figure, the dashed represents the DOMAIN server in which the collection manager and the queue managers execute.

Figure 38.1 Basic Structure of Indirect-Messaging



Queue Management

The queue manager is responsible for allocating the queues, maintaining access information for each of the queues, and administering the messages that belong to each queue.

Queues can be designated as permanent or temporary. Permanent queues remain until they are explicitly deleted. Temporary queues are implicitly deleted when they are closed.

Messages within permanent queues may be persistent or non-persistent. Persistent messages remain indefinitely until they are fetched, even if the DOMAIN server process is shutdown. Non-persistent messages are purged when a queue is closed.

The queue manager is solely responsible for maintaining the queues and for ensuring that the messages in the queues reach their destination when requested and are not lost. The queue manager is also responsible for establishing the information that is needed by the network protocols that are being used to transmit the messages to and from the queues.

Example

The COLLECTION option must be specified in the PROC DOMAIN statement, in order to use indirect messaging. An example follows:

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

After it is started, the collection manager continues to run until the PROC ADMIN statement is used to terminate it.

DOMAIN Registry

Administrator capabilities are an important part of message queueing functionality. Allowing an administrator to register queues provides centralized control of queue definition (how the queue functions as well as who can access it).

All queues are registered or defined, either dynamically or explicitly, through a registration process. A queue that is explicitly registered is known as an *administrator pre-defined* queue. This type of queue is a permanent queue. It can only be deleted by an administrator who uses the Administrator Procedure Interface. This section discusses how to define such a queue. Refer to either Chapter 35, “SAS Component Language (SCL) Interface to Indirect Messaging,” on page 327 or Chapter 37, “CALL Routine Interface to Indirect Messaging,” on page 375 for information about how to dynamically create a queue.

An administrator pre-defined queue can be registered during the PROC DOMAIN collection initialization if proper steps are taken. If a registration file has been created and a fileref of REGISTRY exists that references this file, the DOMAIN server will parse, interpret, and process this registry information.

Registry Syntax

Comments (/* ... */) can be included anywhere within the file. Queue definition as well as administrator privilege registration can be accomplished from four types of directives: USER, GROUP, QUEUE, and ADMIN.

USER Directive - defines users and their identifiers.

Syntax Notes:

username
is a descriptive name for the userid. This name may be a maximum of 32 characters in length. It is only significant to the administrator of this file.

userid
identifies connecting users and may be a maximum of 20 characters in length.

```
USER username userid;
USER username userid;
USER username userid;
```

GROUP Directive - defines a group of users.

Syntax Notes:

groupname
identifies a group of users and may be a maximum of 32 characters in length.

username
identifies a previously defined user name from a USER directive.

```
GROUP groupname
    username
    username
    username;
```

QUEUE Directive - defines message queue and its attributes.

Syntax Notes:

queue_name
identifies the name of the queue and may be a maximum of 32 characters in length.

collection_name
identifies the collection that the queue belongs to. It can be a maximum of 32 characters in length.

msgpsist
identifies whether messages on this queue are persistent. Persistent messages remain on the queue across queue open-and-close boundaries as well as DOMAIN server start-and-stop boundaries. Valid values for this attribute are **yes** or **no**. The default is **no**, which means that messages do not persist.

msgdlvmode
identifies the type of delivery mode operation that is used to fetch or to browse messages from this queue. Valid values for this attribute are **default** (fetch-mode operation) or **notice** (notice-mode operation). The default is **default**, which specifies the fetch-mode operation.

maxdepth
identifies the maximum depth restriction imposed on this queue. Valid queue depth values are in the range **-1**<=*maxdepth*<=*maxint*. The default is **-1**, which indicates unlimited depth.

maxmsgl
identifies the maximum message length for messages that are delivered to this queue. This maximum length must account for additional internal bytes that are needed to represent the data within each message. Attachment lengths are not taken into consideration, only the length of the actual message itself. Valid values are in the range -1<=*maxmsgl*<=*maxint*. The default is **-1**, which indicates unlimited message length.

username_or_groupname
identifies a previously defined user name from a USER directive, a previously defined group name from a GROUP directive, or the special name ANONYMOUS, which can be used to grant privileges to all users.

permissions
are specified by using the following values (an abbreviated form of these values is also acceptable):

DELIVER | D
Deliver privileges.

FETCH | F
Fetch privileges.

BROWSE | B
Browse privileges.

GETPROP | GP
Get property privileges.

SETPROP | SP
Set property privileges.

GETSEC | GS
Get security privileges.

SETSEC | SS
Set security privileges.

ALL
Full privileges.

Combinations of the above are also acceptable by separating values with either a plus sign (**+**) or a comma (**,**). For example, you could give DELIVER and BROWSE privileges to a user by specifying one of the following:

- DELIVER,BROWSE
- D+B (abbreviated form).

replace
identifies the action to take if queue already exists. Valid values for this action are:

no
indicates to continue without replacement (definition is ignored). This is the default.

yes
indicates to replace queue with new definition and refresh the queue by deleting any old messages.

prompt
specifies that you are prompted for the action to take.

```
QUEUE queue_name COLLECTION|C(collection_name)
   MSGPSIST(msgpsist)
   MSGDLVMODE(msgdlvmode)
   MAXDEPTH(maxdepth)
   MAXMSGL(maxmsgl)
   PRIVILEGES(username_or_groupname=permissions
              username_or_groupname=permissions
              username_or_groupname=permissions)
   REPLACE(replace)
;
```

ADMIN Directive - defines queue administrative privileges for a specific user or a group.

Syntax Notes:

username_or_groupname
identifies a previously defined user name from a USER directive, a previously defined group name from a GROUP directive, or the special name ANONYMOUS, which can be used to grant privileges to all users.

permissions
are specified with the following values (an abbreviated form of these values is also acceptable):

UNLIMITED | U
unlimited privileges; user can issue any administrator command.

DISPLAY | D
display privileges; user can only issue the display command. They cannot issue any destructive commands.

replace
identifies the action to take if administrator privileges already exist. Valid values for this action are:

no
indicates to continue without replacement (new privileges are ignored). This is the default.

yes
indicates you should replace existing privileges with new ones.

prompt
specifies that you are prompted for the action to take.

```
ADMIN
   PRIVILEGES(username_or_groupname=permissions
              username_or_groupname=permissions
              username_or_groupname=permissions)
   REPLACE(replace)
;
```

Agent Scheduling

Agent services is used with compute services and messaging services to provide client/server based task management for the nodes across your network. An *agent* is simply SAS source code that is launched by the DOMAIN server to support the execution of a task on a remote node. After the task has finished executing, an agent may be designed to send a *completion notification* to a message queue for the client application, or the client can check the DOMAIN server to see if the agent has completed execution.

The source that composes an agent may be pre-defined and stored with the DOMAIN server, or the source may be defined dynamically with the request for the agent to execute.

Client/server-based agents provide the following types of services:

- distributed agent processing
- periodic agent processing
- conditional agent processing
- parallel agents processing.

These services can be used together to maximize the flexibility and functionality of a client/server application. For example, using agent services, an application can be designed to execute autonomously on a periodic basis. The application can make conditional decisions to satisfy its goals, including submitting other processes for execution on various hosts on the network. See Chapter 40, “Using Agent Services,” on page 429 for more information.

The AGENT option must be specified in the PROC DOMAIN statement, in order to use agent services. An example follows:

```
libname domain ".";
proc domain agent;
run;
```

For more information about configuring and using the DOMAIN server, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software.*

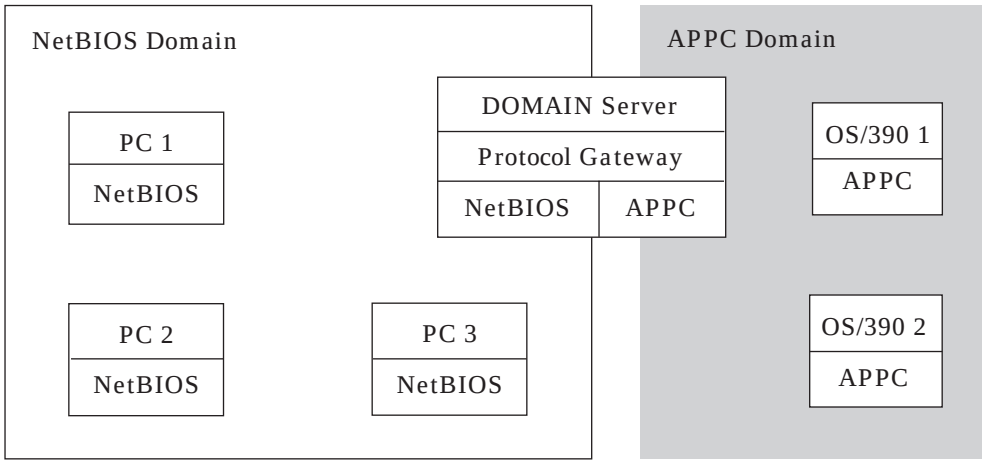
Protocol Gateway

The protocol gateway service connects two SAS sessions that do not use a common communications access method in a networked environment. A network comprises one or more logical domains, each of which maps a topological area according to the communications protocol or access method that it supports.

A SAS session can communicate with any other SAS session that runs in the same logical domain because both sessions use the same access method. Furthermore, SAS sessions from different logical domains can communicate by means of the DOMAIN server if at least one domain supports multiple access methods (one of which must be used in both domains). The DOMAIN server eliminates the need to configure multiple protocols in a given domain.

The DOMAIN server provides the gateway that allows communication between SAS sessions that are running in different logical domains, as illustrated in the following figure.

Figure 38.2 Protocol Gateway Service with DOMAIN Server



All sessions in both domains can intercommunicate by using the DOMAIN server's set of common access methods. A message flows over the originator's native access method to the DOMAIN server. This access method is called the inbound access method. The server re-directs a message to the cross-domain destination by using the destination's native access method. Conversely, the outbound access method is the access method that is used on the target host side of the protocol gateway.

System Requirements

The DOMAIN server requires a dedicated SAS session that runs in an OS/2 or in a Windows NT environment.

Using the Protocol Gateway

To connect two SAS sessions when using the protocol gateway of the DOMAIN server, you must initialize the DOMAIN server and specify the inbound and outbound communication access methods for the DOMAIN server. For the TCP/IP outbound access method, sign-on scripts must be specified for the DOMAIN server.

Note: If you use the TCP/IP access method, you may need to configure the DOMAIN server in the SERVICES file. △

In addition, you must set macro variables at the SAS/CONNECT local host or at the SAS/SHARE client to identify the DOMAIN server node name and the server identifier and to provide DOMAIN server security for connecting local hosts or clients. The macro variables that are set are based on the inbound access method that is used.

You must also specify the remote host that you are connecting to, the local host, and the security environment variables that will be used for the userid and password of the target host. The DOMAIN server supplies the specified value to the outbound access method, which negotiates security with the target host.

For more information about configuring and using the DOMAIN server, see *Communications Access Methods for SAS/CONNECT and SAS/SHARE Software*.

Example

The PROTOCOL option in the PROC DOMAIN statement is used to create the DOMAIN server DOMSVR using the TCP/IP access method. The COMAMID, COMAUX1 and COMAUX2 options should be specified on the command line or in your configuration file.

The macro variable GWHOST stores the fully qualified TCP/IP node name of the DOMAIN server STAR.XYZ.ABC and has the userid BASS and the password TIME2GO.

```
proc domain protocol serverid=domsrv;
```

Submit the following statements from the SAS/CONNECT local host:

```
%let GWHOST=STAR.XYZ.ABC;
%let TCPGW=GWHOST.DOMSRV;
%let TCPSEC=bass.time2go;

options comamid=tcp remote=RMTNODE;
signon;
```

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