Remote Procedure Call (RPC) Configuration

This chapter provides background and guidelines to help you configure the MAPCFGxx member of the PARM data set. MAPCFGxx sets up the Portmapper for Remote Procedure Calls (RPC). This chapter includes these sections:

- Introduction to the Portmapper

  Describes the operation of the portmapper

- MAPCFGxx Configuration

  Describes the MAPCFGxx configuration member.

Introduction to the Portmapper

RPC-based client programs need a way to find RPC-based server programs, or to look up and find the port numbers of server programs. The naming of services by way of the port number segment of their IP address is mandated by the Internet protocols. Given this, clients face the problem of determining which ports are associated with the services they want to use.

The MAP task group is responsible for mapping Remote Procedure Call (RPC) programs and version numbers to transport specific port numbers. This service is required by NFS and application programs using the RPC interface of the Cisco IOS for S/390 API.

The port mapper protocol defines an RPC network service that provides a standard way for RPC-based clients to look up the port number of any remote RPC-based program supported by a server. Because it can be implemented on any transport that provides the equivalent ports, it provides a single solution to a general problem that works for all clients, all servers, and all networks.

Port Registration

Every port mapper on every host is associated with port number 111. The port mapper is the only RPC network service that must have such a well known (dedicated) port. Other network services can be assigned port numbers statically or dynamically as long as they register their ports with the host port mapper.

For example, an RPC-based server program typically gets a port number at runtime by calling an RPC library procedure. Note that a given network service can be associated with port number 256 on one server and with port number 885 on another. On a given host, a service can be associated with a different port every time its server program is started.

Delegating port-to-remote program mapping to port mappers also automates port number administration. Statically mapping ports and remote programs in a file duplicated on each client requires updating all mapping files whenever a new remote program is introduced to a network. (The alternative of placing the port-to-program mappings in a shared NFS file would be too centralized, and if the file server went down, the whole network would go down with it.)

The collection of port-to-program mappings that are maintained by the port mapper server is called a port map. To make the port mapper start automatically whenever Cisco IOS for S/390 is started, place the START MAP command in the START00 member of the PARM data set.

As shown in the typical port mapping sequence here, both RPC server programs and client programs call port mapper procedures.

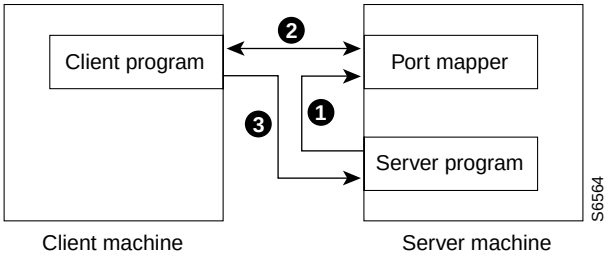
Note Although RPC client and server programs and client and server machines are usually distinct, they need not be. A server program can also be a client program, as when an NFS server calls a port mapper server. Likewise, when a client program directs a remote procedure call to its own machine, the machine acts as both client and server.

As part of its initialization, an RPC server program calls the host port mapper to create a port map entry. Whereas server programs call port mappers to update port map entries, clients call port mappers to query port map entries.

To find the port of a remote program, a client sends an RPC call message to the server port mapper. If the remote program is supported on the server, the port mapper returns the associated port number in an RPC reply message. The client program can then send RPC call messages to the port of the remote program. A client program can minimize its port mapper calls by caching the port numbers of recently called remote programs.

The port mapper provides an inherently stateful service because a port map is a set of associations between registrants and ports.

Figure 14-1 Typical Port Mapping Sequence.



Step 1 Server registers port with the port mapper.

Step 2 Client gets server's port from port mapper.

Step 3 Client calls server's port directly.

The port mapper protocol provides the procedure, callit(), by which the port mapper can assist a client in making a remote procedure call. A client program passes the program number, version number, procedure number and arguments of the target procedure in a call message. callit() looks up the port number of the target procedure in the port map and sends a call message to the target procedure which contains the arguments received from the client.

When the target procedure returns results to callit(), callit() returns the results to the client program. It also returns the port number of the target procedure so the client can subsequently call the target procedure directly.

Because every instance of a remote program can be mapped to a different port on every server, a client has no way to broadcast a remote procedure call directly. However, the port mapper callit() procedure can be used to broadcast a remote procedure call indirectly, since all port mappers are associated with port number 111.

One way for a client to find a server running a remote program is to broadcast a call to callit(), asking it to call procedure 0 of the desired remote program. If this call is broadcast to all servers, the first reply received is likely to be from the server with the lightest work load.

MAPCFGxx Configuration

The MAPCFGxx member in the PARM data set specifies the configuration parameters for the MAP task group.

To specify a MAPCFGxx member other than the default MAPCFG00, specify CNFG=xx in the START MAP command found in the START00 member of the PARM data set.

PORTMAP Statement Syntax

PORTMAP [APISUBSYS (*subsystem_name***)] [DEBUG | NODEBUG]**

Syntax Description

APISUBSYS (*subsystem_name*)	Specifies the MVS subsystem name of the API access method subsystem. The special name of **** indicates that the API subsystem resides in the same address space with the MAP task group. Default: ****
DEBUG | NODEBUG	Specifies whether the port mapper writes debug trace records to the MAPLOG and MAPERR DD data sets for each port map request it receives or reply it sends. Default: NODEBUG

Example

This example shows the usage of the PORTMAP statement:

```
*-------------------------------------------------------------*
*      SPECIFY PORT MAPPER START UP PARAMETERS
*-------------------------------------------------------------*
PORTMAP APISUBSYS(****)
        NODEBUG
```