Ex No : 11 A **Date :7/11/2024**

STUDY OF REMOTE PROCEDURE CALL- XMLRPC

AIM:

To study the concepts of Remote Procedure Call-XML RPC.

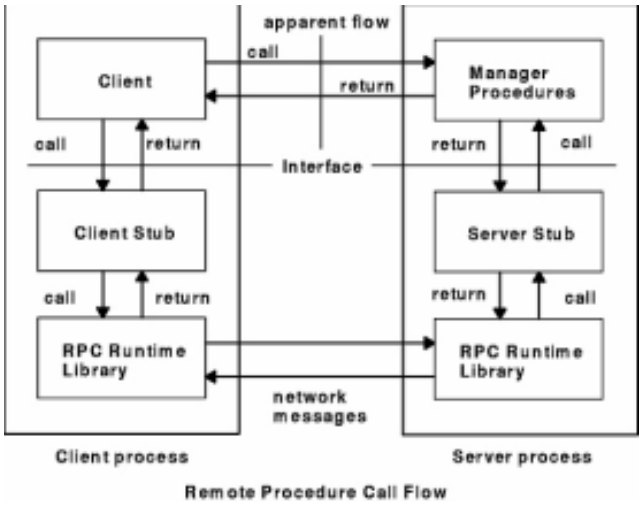
What is RPC programming?

Remote Procedure Call (RPC) programming allows a program to call procedures outside its current address space. A program can execute a procedure on a remote machine, pass data to it, and retrieve the result. Using RPC, you can write a distributed

INTRODUCTION TO RPC:

- Remote Procedure Call (RPC) is a model that specifies how cooperating processes on different nodes in a heterogeneous computing environment can communicate and coordinate activities.
- The paradigm of RPC is based on the concept of a procedure call in a higher level programming language.
- The semantics of RPC are almost identical to the semantics of the traditional procedure call.
- The major difference is that while a normal procedure call takes place between procedures of a single process in the same memory space on a single system, RPC takes place between processes on clients and servers in a heterogeneous computing environment.
- The remote procedure call act like a procedure call, but act across the network. • The process makes a remote procedure call by pushing its parameters and a return address onto the stack, and jumping to the start of the procedure. The procedure itself is responsible for accessing and using the network.
- After the remote execution is over, the procedure jumps back to the return address. The calling process then continues.

RPC WORKING MODEL:



Remote Procedure Call Flow

1. The client calls the client stub. The call is a local procedure call, with parameters pushed on to the

stack in the normal way.

2. The client stub packs the parameters into a message and makes a system call to send the message. Packing the parameters is called marshaling.
3. The client's local operating system sends the message from the client machine to the server machine.
4. The local operating system on the server machine passes the incoming packets to the server stub.
5. The server stub unpacks the parameters from the message. Unpacking the parameters is called unmarshalling.
6. Finally, the server stub calls the server procedure. The reply traces the same steps in the reverse direction.

- When the calling process calls a procedure, the action performed by that procedure will not be the actual code as written, but code that begins network communication. It has to connect to the remote machine, send all the parameters down to it, wait for replies, do the right thing to the stack and return. This is the **client side *stub***.
- **The server side stub** has to wait for messages asking for a procedure to run. It has to read the parameters, and present them in a suitable form to execute the procedure locally. After execution, it has to send the results back to the calling process.

BASIC PROCESS OF A SERVER:

RPC BINDING

• Server program defines the server's interface using an interface definition language(IDL)• The IDL specifies the names, params, and type for all server procedures • A stub compiler reads the IDL and produces two stub procedures for each server procedure: a client-side stub and a server-side stub
• The server writer writes the server and links it with the server-side stubs; the client writes the program and links it with the client-side stubs.

RPC STUBBING

• A client-side stub looks to the client as if it were a callable server procedure • A server-side stub looks to the server as if it were a calling client
• The client program thinks it is calling the server, in fact it calling the client stub • The server program thinks it is called by client, in fact, it is called by the server stub• The stubs send messages to each other to make the RPC happen.

RPC MARSHALING

• Marshalling is the packing of procedure parameters into a message packet. • The RPC stubs call procedures to marshal (or unmarshal) all parameters. • On the client side, the client stub marshals the parameters into the call packet; on the
server side the server stub unmarshals the parameters in order to call the server's procedure
• On the return, the server stub marshals return parameters into the return packet; the client stub unmarshals return parameters and returns to the client.

XML-RPC METHOD IN PYTHON

INTRODUCTION:

XML-RPC is a lightweight remote procedure call protocol built on top of HTTP and XML. With it, a client can call methods with parameters on a remote server (the server is named by a URI) and get back structured data.

In PYTHON, ***xmlrpc*** is a package that collects server and client modules implementing XML RPC. The modules are:

- xmlrpc.client
- xmlrpc.server

xmlrpc.client: This module supports writing XML-RPC client code; it handles all the details of translating between conformable Python objects and XML on the wire.

METHODS:

class **xmlrpc.client.ServerProxy**(*uri*, *transport=None*, *encoding=None*, *verbose=False*, *allow_none=False*, *use_datetime=False*, *use_builtin_types=False*, *****, *context=None*)

- ServerProxy instance is an object that manages communication with a remote XML-RPC server.
- The required first argument is a URI (Uniform Resource Indicator), and will normally be the URL of the server. All other arguments are optional.
- The returned instance is a proxy object with methods that can be used to invoke corresponding RPC calls on the remote server.

xmlrpc. server:

This module provides a basic server framework for XML-RPC servers written in Python. Servers can either be free standing, using SimpleXMLRPCServer, or embedded in a CGI environment, usingCGIXMLRPCRequestHandler.

METHODS:

class xmlrpc.server.SimpleXMLRPCServer(addr, requestHandler=SimpleXMLRPCRequestHandler, logRequests=True, allow_none=False, encoding=None, bind_and_activate=True, use_buil tin_types=False)

- Create a new server instance.
- This class provides methods for registration of functions that can be called by the XML RPC protocol.
- The requestHandler parameter should be a factory for request handler instances; it defaults to

SimpleXMLRPCRequestHandler.

- The addr and requestHandler parameters are passed to the socketserver.TCPServer constructor.
- If logRequests is true (the default), requests will be logged; setting this parameter to false will turn off logging.
- The allow_none and encoding parameters are passed on to xmlrpc.client and control the XML-RPC responses that will be returned from the server.
- The bind_and_activate parameter controls whether server_bind() and server_activate() are called immediately by the constructor; it defaults to true. Setting it to false allows code to manipulate the allow_reuse_address class variable before the address is bound.
- The use_builtin_types parameter is passed to the loads() function and controls which types are processed when date/times values or binary data are received; it defaults to false.

SimpleXMLRPCServer Objects

The SimpleXMLRPCServer class is based on socketserver.TCPServer and provides a means of creating simple, stand alone XML-RPC servers.

SimpleXMLRPCServer.register_function(*function=None*, *name=None*) Register a function that can respond to XML-RPC requests. If *name* is given, it will be the method name associated with *function*, otherwise function.__name__ will be used. *name* is a string, and may contain characters not legal in Python identifiers, including the period character.

This method can also be used as a decorator. When used as a decorator, *name* can only be given as a keyword argument to register *function* under *name*. If no *name* is given, function.__name__ will be used.