

Quick Specs:

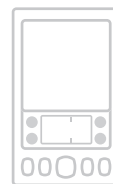
Category: Web Services

Desktop / Server OS: Windows, UNIX, Linux (anywhere ProvideX runs)

Requires ProvideX web server for server components

Who it's for:

Any ProvideX (PVX) developer who needs to build more complicated systems to allow any number or type of client devices to interact with ProvideX at run-time.



Why Does RemoteXerver Exist?

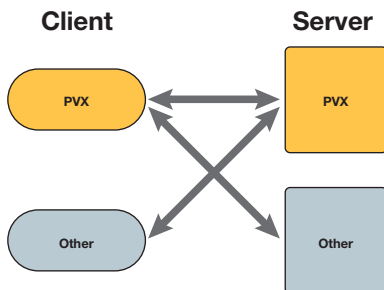
More and more, we are relying on applications that communicate over the Internet. These applications consist of one program that makes a procedure call to another program. Since the two programs may be on different computers, or even on different systems on opposite ends of the world, this call is referred to as a Remote Procedure Call (RPC). There are two parts to the RPC – a client side application and a server side application. The client side initiates the RPC, and the server side executes the desired procedure and returns the results.

RemoteXerver is an implementation of XML-RPC. XML-RPC is one of many ways to perform RPCs, but is gaining wide acceptance because it uses a standardized XML document to hold the request and response for the RPC.

Access ProvideX from Anywhere

RemoteXerver is an XML-RPC framework that enables the creation of ProvideX-based XML-RPC client and server applications. These client and server applications allow programs written in ProvideX running on one system to communicate with programs on another system over the Internet or any IP network. Since XML-RPC is an industry standard, RemoteXerver will allow ProvideX programs to communicate with programs written in other programming languages.

RemoteXerver is designed to manage the details of the communication so that it appears like a regular method or function call. A client application may then be developed without needing to know how the communication is done. Provide the program function name, web address, and parameters, and RemoteXerver takes care of all the communication details and returns the result to you. A server application may also be written for RemoteXerver without concern for the communication details.



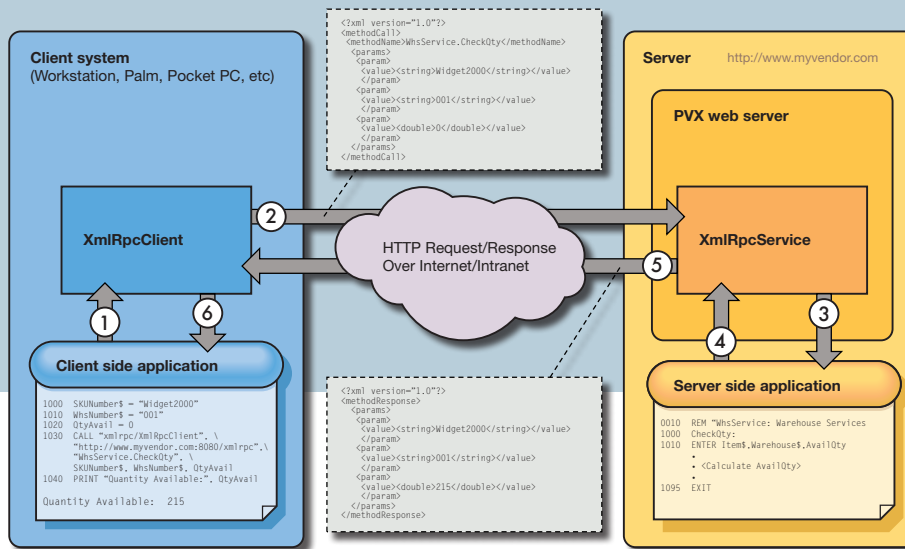
RemoteXerver implements both the client and the server sides. That way you can have any combination of ProvideX programs and other applications.

See page 2 for examples and diagram



Development Partner

GOLD



How Does It Work?

Step 1: On the client side, when a program needs to communicate to the server, it passes the request to the XmlRpcClient.

Step 2: The XmlRpcClient then wraps up the information and packages it in an XML document, which is sent over the Internet to the server.

Step 3: The XmlRpcService running on the server then reads this XML document, figures out what the request is, and passes the information to the desired program.

Step 4: The program returns its results to the XmlRpcService.

Step 5: The XmlRpcService packages the results in another XML document and returns it to the client.

Step 6: The XmlRpcClient reads the XML response, figures out the results, and passes them to the original program.

A Few Examples of What You Can Do With RemoteXerver:

Example 1: A developer needs to access ProvideX data from a handheld device at run-time. The developer locates an XML-RPC client for the handheld device and proceeds to develop the handheld device's application. When the developer needs to access ProvideX data, a call is made to the ProvideX server running RemoteXerver, using the handheld device's XML-RPC client, to retrieve the data.

Example 2: A developer needs to create a ProvideX program to allow remote access to a ProvideX server. In this example, RemoteXerver is used on both the client and the server.

Example 3: A developer needs to create a solution to access vendor data using XML-RPC. Just the RemoteXerver client is necessary, as the vendor is running an XML-RPC capable server. The vendor provides the server's URL, available functions and their corresponding parameters. The developer then uses that information in conjunction with RemoteXerver client to retrieve data from the vendor's server.