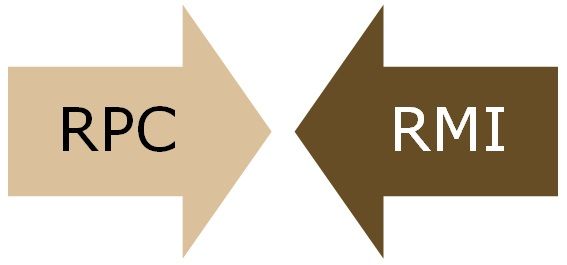
**RPC** stands for **Remote Procedure Call** which supports procedural programming. It’s almost like an IPC mechanism wherever the software permits the processes to manage shared information Associated with an environment wherever completely different processes area unit death penalty on separate systems and essentially need message-based communication.

The above diagram shows the working steps in RPC implementation. **RMI** stands for **Remote Method Invocation**, which is a similar to RPC but it supports [object-oriented programming](https://www.geeksforgeeks.org/object-oriented-programming-oops-concept-in-java/) which is java’s feature. A thread is allowable to decision the strategy on a foreign object. In RMI, objects are passed as a parameter rather than ordinary data.

This diagram shows the client-server architecture of the RMI protocol. RPC and RMI both are similar but the basic difference between RPC and RMI is that RPC supports procedural programming, on the other hand, RMI supports object-oriented programming. Let’s see that the difference between RPC and RMI:

| **S.NO** | **RPC** | **RMI** |
| --- | --- | --- |
| 1. | [RPC](https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system/) is a library and OS dependent platform. | Whereas it is a java platform. |
| 2. | RPC supports procedural programming. | [RMI](https://www.geeksforgeeks.org/remote-method-invocation-in-java/) supports object-oriented programming. |
| 3. | RPC is less efficient in comparison of RMI. | While RMI is more efficient than RPC. |
| 4. | RPC creates more overhead. | While it creates less overhead than RPC. |
| 5. | The parameters which are passed in RPC are ordinary or normal data. | While in RMI, objects are passed as parameter. |
| 6. | RPC is the older version of RMI. | While it is the successor version of RPC. |
| 7. | There is high Provision of ease of programming in RPC. | While there is low Provision of ease of programming in RMI. |
| 8. | RPC does not provide any security. | While it provides client level security. |
| 9. | It’s development cost is huge. | While it’s development cost is fair or reasonable. |
| 10. | There is a huge problem of versioning in RPC. | While there is possible versioning using RDMI. |
| 11. | There is multiple codes are needed for simple application in RPC. | While there is multiple codes are not needed for simple application in RMI. |
| 12. | Can be complex due to low-level implementation | Generally simpler to use and implement |

Difference Between RPC and RMI

RPC and RMI are the mechanisms which enable a client to invoke the procedure or method from the server through establishing communication between client and server. The common difference between RPC and RMI is that RPC only supports **procedural programming** whereas RMI supports **object-oriented programming**.

Another major difference between the two is that the parameters passed to remote procedures call consist of **ordinary data structures**. On the other hand, the parameters passed to remote method consist of **objects**.

Content: RPC Vs RMI

1. [Comparison Chart](https://techdifferences.com/difference-between-rpc-and-rmi.html#ComparisonChart)
2. [Definition](https://techdifferences.com/difference-between-rpc-and-rmi.html#Definition)
3. [Key Differences](https://techdifferences.com/difference-between-rpc-and-rmi.html#KeyDifferences)
4. [Conclusion](https://techdifferences.com/difference-between-rpc-and-rmi.html#Conclusion)

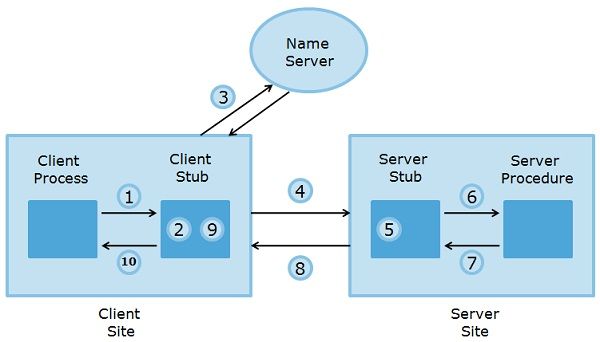
Comparison Chart

| **BASIS FOR COMPARISON** | **RPC** | **RMI** |
| --- | --- | --- |
| Supports | Procedural programming | Object-oriented programming |
| Parameters | Ordinary data structures are passed to remote procedures. | Objects are passed to remote methods. |
| Efficiency | Lower than RMI | More than RPC and supported by modern programming approach (i.e. Object-oriented paradigms) |
| Overheads | More | Less comparatively |
| In-out parameters are mandatory. | Yes | Not necessarily |
| Provision of ease of programming | High | low |

Definition of RPC

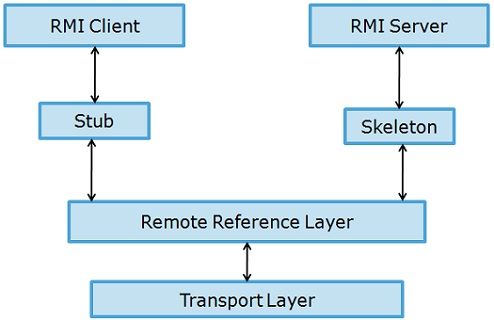
**Remote Procedure Call (RPC)** is a programming language feature devised for the distributed computing and based on semantics of**local procedure** calls. It is the most common forms of remote service and was designed as a way to abstract the procedure call mechanism to use between systems connected through a network. It is similar to IPC mechanism where the operating system allows the processes to manage shared data and deal with an environment where different processes are executing on separate systems and necessarily require message-based communication.

Let’s understand how RPC is implemented through the given steps:

* The client process calls the client stub with parameters, and its execution is suspended until the call is completed.
* The parameters are then translated into machine-independent form by marshalling through client stub. Then the message is prepared which contain the representation of the parameters.
* To find the identity of the site the client stub intercommunicate with name server at which remote procedure exists.
* Using blocking protocol the client stub sends the message to the site where remote procedure call exists. This step halt the client stub until it gets a reply.
* The server site receives the message sent from the client side and converts it into machine specific format.
* Now server stub executes a call on the server procedure along with the parameters, and the server stub is discontinued till the procedure gets completed.
* The server procedure returns the generated results to the server stub, and the results get converted into machine-independent format at server stub and create a message containing the results.
* The result message is sent to the client stub which is converted back into machine specific format suitable for the client stub.
* At last client, stub returns the results to the client process.

Definition of RMI

**Remote Method Invocation (RMI)** is similar to RPC but is language specific and a feature of java. A thread is permitted to call the method on a remote object. To maintain the transparency on the client and server side, it implements remote object using stubs and skeletons. The stub resides with the client and for the remote object it behaves as a proxy.

When a client calls a remote method, the stub for the remote method is called. The client stub is accountable for creating and sending the parcel containing the name of a method and the marshalled parameters, and the skeleton is responsible for receiving the parcel.The skeleton unmarshals parameters and invokes the desired method on the server. The skeleton marshals the given value (or exceptions) with the parcel and sends it to client stub. The stub reassembles the return parcel and sends it to the client.

In Java, the parameters are passed to methods and returned in the form of reference. This could be troublesome for RMI service since not all objects are possibly remote methods. So, it must determine which could be passed as reference and which could not.

Java uses process named as **serialisation** where the objects are passed as value. The remote object is localised by pass by value. It can also pass an object by reference through passing a remote reference to the object along with the URL of the stub class. Pass by reference restricts a stub for the remote object.

Key Differences Between RPC and RMI

1. RPC supports procedural programming paradigms thus is C based, while RMI supports object-oriented programming paradigms and is java based.
2. The parameters passed to remote procedures in RPC are the ordinary data structures. On the contrary, RMI transits objects as a parameter to the remote method.
3. RPC can be considered as the older version of RMI, and it is used in the programming languages that support procedural programming, and it can only use pass by value method. As against, RMI facility is devised based on modern programming approach, which could use pass by value or reference. Another advantage of RMI is that the parameters passed by reference can be changed.
4. RPC protocol generates more overheads than RMI.
5. The parameters passed in RPC must be “**in-out**” which means that the value passed to the procedure and the output value must have the same datatypes. In contrast, there is no compulsion of passing “**in-out**” parameters in RMI.
6. In RPC, references could not be probable because the two processes have the distinct address space, but it is possible in case of RMI.

Conclusion

Both RPC and RMI serves the same purpose but are used in languages support different programming paradigms, therefore have distinct features.