

TECHNOLOGY INCUBATION CENTER  
IBM Software Labs, India

# IBM Service Management Framework Extension for Relocatable Services

IBM alphaWorks Technology  
Technical Whitepaper

**VERSION** 0.9.1  
**SHORT NAME** ReSMF  
**KEYWORDS** Relocatable, Services, SMF, Framework

## ABSTRACT

In the “On Demand Era” the enterprise services and applications have to be receptive and responsive to the local as well as the global state of their operating environment. They need to respond automatically to the changes in their state and need to work around problems, security threats, and system failures. These services are often composed dynamically from multiple service end-points. Therefore the availability and scalability of the service end-points become critical if the composed service has to meet its Quality of Service (QoS) commitments. The availability of the service end-points in turn depend on the robustness of their hosting environment. When the service hosting environment poses impediments to their smooth running, the services fail in their QoS commitments on availability and in turn impact the QoS of the consuming enterprise applications. For the better availability and scalability of services, they should be made relocatable or mobile in other words. A relocatable service/application should be able to move autonomously along with its state to another identical hosting environment in the event of adverse conditions in the current environment or if user wants it to do so. Similarly, to improve scalability in a dynamic fashion, a relocatable service should be able to clone itself onto identical environments in the adverse situations. The validity and feasibility of this proposition is examined by taking IBM Service Management Framework (SMF) as the test bed in our work called as *IBM Service Management Framework Extension for Relocatable Services (ReSMF)*.

**HOMEPAGE** <http://www.alphaworks.ibm.com/tech/resmf>  
**LICENSE** <http://www.alphaworks.ibm.com/license>

**AUTHORS**

Manu Kuchhal  
[manukuchhal@in.ibm.com](mailto:manukuchhal@in.ibm.com)

Umesh Joshi  
[joshi.umesh@in.ibm.com](mailto:joshi.umesh@in.ibm.com)

Rohit M Singh  
[rohitms@in.ibm.com](mailto:rohitms@in.ibm.com)

Umasuthan Ramakrishnan  
[umasuthan@in.ibm.com](mailto:umasuthan@in.ibm.com)

**PROJECT LEAD** Albee Jhoney  
[albee.jhoney@in.ibm.com](mailto:albee.jhoney@in.ibm.com)  
Group Leader,  
Technology Incubation Center  
IBM Software Labs, India

## Table of Content

<b>INTRODUCTION</b> .....	<b>3</b>
DEFINITIONS & ACRONYMS.....	3
<b>MOTIVATION</b> .....	<b>3</b>
<b>OBJECTIVE</b> .....	<b>4</b>
<b>SMF EXTENSION FOR RELOCATABLE SERVICES</b> .....	<b>4</b>
MOBILE SERVICES FRAMEWORK (MOSEF).....	4
MOSEF CLIENT API .....	6
<i>Working with MSBs</i> .....	6
MOSEF ADMINISTRATION CONSOLE .....	6
<b>MOBILE SERVICES FRAMEWORK - FEATURES</b> .....	<b>8</b>
SERVICE MOBILITY PRIMITIVES.....	8
<i>Dispatch Operation</i> .....	8
<i>Clone Operation</i> .....	8
<i>Retract Operation</i> .....	8
<i>Fetch Operation</i> .....	8
MESSAGING PRIMITIVES .....	8
<i>Messaging between two MSBs</i> .....	8
DATA VIRTUALIZATION.....	9
SERVICE LOCATION TRANSPARENCY.....	9
STATEFUL APPLICATION RELOCATION .....	11
<b>CONCLUSION</b> .....	<b>11</b>
<b>ACKNOWLEDGEMENTS</b> .....	<b>11</b>
<b>RESOURCES</b> .....	<b>11</b>

## Introduction

This is a technical white paper that describes the need for an availability and scalability of services and briefly explains one mechanism for achieving the same in IBM Services Management Framework.

## *Definitions & Acronyms*

Here are the definitions of some of the terms and acronyms used in this document.

SOA	Service Oriented Architecture
OSGi	Open Service Gateway Initiative
SMF	<b>Service Management Framework</b> is an IBM implementation of OSGi
Relocatable Service	A <b>Relocatable Service</b> is a service that can be moved or cloned autonomously along with its state to another identical hosting environment in the event of adverse conditions in the current environment.
MoSeF	<b>Mobile Services Framework</b> provides a convenient environment for the hosting and the management of the relocatable services.
MSB	<b>Mobile Service Bundle</b> is a logical entity that refers to a packaged relocatable service.

## Motivation

In order to stay focused and enhance the responsiveness in the On Demand Era, enterprise applications are increasingly modeled after the Service Oriented Architecture (SOA). In SOA the enterprise applications are implemented as services that are composed of multiple service-endpoints. Here, the QoS commitments of the enterprise application are tightly bound to the availability and scalability of the composed service end-points. There are numerous clustering and load-balancing technologies prevalent that provide the availability and scalability of services. However they require cluster to be defined before hand and the services need to be pre-installed on each node that is going to be a part of the cluster. Take the case when a hosting infrastructure is being upgraded and the existing infrastructure is to be phased out. In this case the services must seamlessly migrate, state-fully from the existing hosting environments to the new ones without disrupting business. Another scenario where real time migration of service would be required is over a Grid. Here the Grid Services may want to migrate from one node to another, due to the fact that their initial contract with the host has expired, or the host node's performance is deteriorating, or the node is going to move away from the network, or the services may like to multiply in the identical environments to improve availability to the client.

To be able to scale up on demand and ensure high availability and scalability, services should become virtualized with respect to their hosting environment. To be more specific, services should be: -

- Capable of finding other suitable and identical hosting environments in a defined domain
- Capable of gracefully shutting down from one hosting environment and moving to another suitable environment along with their state.
- Capable of being cloned to another suitable hosting environment along with their state..
- Transparent to the consumer (of that service) as far as their location is concerned. This implies that even if the services might have moved to the new location (host), the consumer is able to work with them.

To facilitate such autonomous behavior of services more completely, service hosting environments must provide **relocatability** as an additional feature to the services, The primitives like **move** and **clone need to be supported** apart from the typical **start, stop, suspend** and **stop** applicable over hosted services.

## Objective

The primary objective is to testify the role of relocatability in improving the availability and scalability of services.

To enable the management of availability and scalability autonomously, the services need to be made relocatable. It should be possible to move/clone the services from one node to another. The services should become receptive to the events taking place in their local and global environments and be able to respond to them by moving or cloning themselves to other suitable hosting environments along with their current state. All of this has to be accomplished by designing an application framework that extends from the primitives provided by the existing service framework and must smoothly plug into it.

Another objective is to provide an easy and comprehensive model for programming the new relocatable services and bringing in relocatability as an add-on feature to the existing services.

It is also desirable to have a Remote Administration Console through which administrators can monitor the movement of bundles and perform actions that affect their lifecycle and relocation.

## SMF Extension for Relocatable Services

IBM Service Management Framework™ (SMF), an implementation of the OSGi™ Service Platform specification, provides for the network delivery and management of applications and services independent of operating system and instruction set architecture (ISA).

SMF provides a layer that allows deploying multiple applications on a single Java Virtual Machine (JVM). Application developers partition applications into services and other resources that are packaged into **bundle**. A bundle serves as the basic delivery unit for applications in SMF. The underlying SMF platform resolves the service dependencies and package dependencies of the bundles automatically.

Typically, applications are composed of a group of service components that interact with each other and other external service components to provide the desired functionality/service. Such a design and runtime model provides the flexibility of deploying value added features to applications by the deployment of additional service bundles.

In addition to multiple services running within the same JVM, this way of deploying and offering the services does not require the JVM to be re-cycled each time a new service is deployed into the environment.

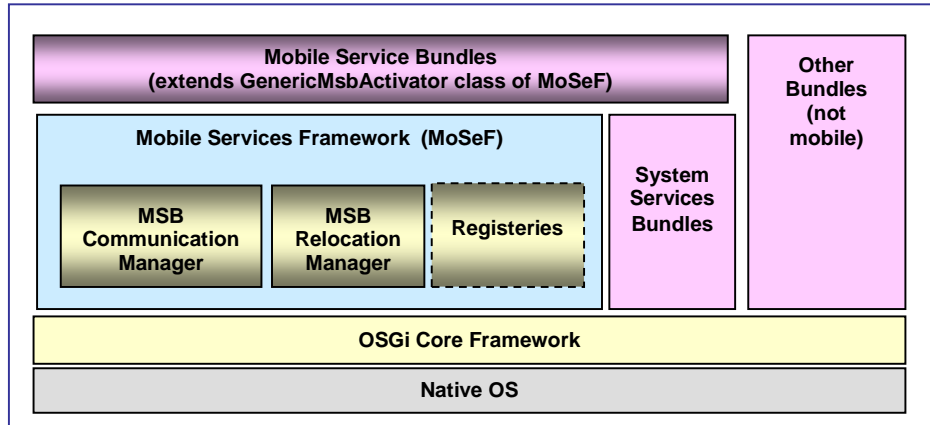
The quest to accomplish the set objectives has resulted in the development of *IBM Service Management Framework Extension for Relocatable Services* comprising of three components: -

- Mobile Services Framework (MoSeF).
- MoSeF Client API.
- MoSeF Administration Console.

### *Mobile Services Framework (MoSeF)*

MoSeF is built on IBM Service Management Framework (SMF) [1] that further strengthens SMF by providing better availability and scalability to the services running in it. MoSeF itself is

deployed as a service bundle within SMF environment. The following is the logical component architecture of MoSeF.



A SMF bundle becomes a **Mobile Service Bundle (MSB)** by extending the *GenericMsbActivator* class, which is provided as a part of MoSeF implementation. This class has various callback methods that are intercepted by the MoSeF container hosted on the same SMF node and takes control of the life-cycle of the MSB. Thus not only these bundles are able to make use of the services provided by the MoSeF, but MoSeF can also act upon them. Lifecycle of a MSB is a super set of the life cycle of a normal bundle i.e. a MSB will have all the possible states that a SMF bundle has, but in addition it can also be in some other well defined states specific to MoSeF context.

There are two ways to create a new instance of a Mobile Service Bundle. The first is to instantiate a completely new Mobile Service Bundle from class definitions/template available either locally or from a remote location. The other way is to create a copy of an existing Mobile Service Bundle by cloning it. The cloned Mobile Service Bundle has the same state as the original one but has a distinct identity of its own. Once created, a Mobile Service Bundle can be dispatched to and/or retracted from a remote location, deactivated and placed in secondary storage, then activated later. And this cycle may continue in forward or backward direction.

MoSeF is the main component which brings relocatability as an add-on feature to the SMF bundles. It comprises of the following sub-components: -

<b>MSB Relocation Manager</b>	Handles the stateful relocation and cloning of MSBs across the MoSeF nodes.
<b>MSB Communication Manager</b>	Enables communications between <ul style="list-style-type: none"> <li>• MSB instances,</li> <li>• MSBs and Registries</li> </ul>
<b>Host Registry</b>	Maintains a record of the currently active MoSeF nodes and their respective. The following operations are supported by the Host Registry:- <ul style="list-style-type: none"> <li>• <b>Register</b> – to register a MoSeF node</li> <li>• <b>Unregister</b> – to unregister a MoSeF node</li> <li>• <b>Query</b> – to query for a matching MoSeF node for a required node profile (property values like CPU, Memory).</li> <li>• <b>Update Property</b> – updates the properties associated with a registered MoSeF node.</li> </ul>

---

<b>Service Registry</b>	Hosts a record of all the services and their location in a MoSeF domain.
-------------------------	--------------------------------------------------------------------------

---

*Note: The SMF host, on which MoSeF is installed, will be hereafter referred to as MoSeF Node / MoseF Host interchangeably.*

## ***MoSeF Client API***

The MoSeF Client API provides the interfaces and classes for client applications to interact with service registry and obtain the current handle to a service.

## **Working with MSBs**

To work with an MSB instance the client must first obtain a valid handle to the service instance from the service registry. During the course of a session with an MSB it is possible that a service handle is invalidated due to the relocation of the service. In this case, there is a service fault. Here the client may wish to continue transactions with another instance of the MSB or with the same instance of the MSB. In either case the client must obtain a fresh service handle from the service registry. However if the client must work with the same instance of the MSB then the previous handle must be passed to the service registry when obtaining a fresh handle.

To obtain a handle to an MSB, the client queries the service registry by passing a Service Descriptor Object. This object contains the ServiceName, QoS parameters, load factor and a set of properties that the MSB must satisfy (optional). Currently only three such properties are supported (OS, Processor type, free memory). The Service Registry, match the list of properties against the required profile to identify the suitable MSB which also has the least load factor.

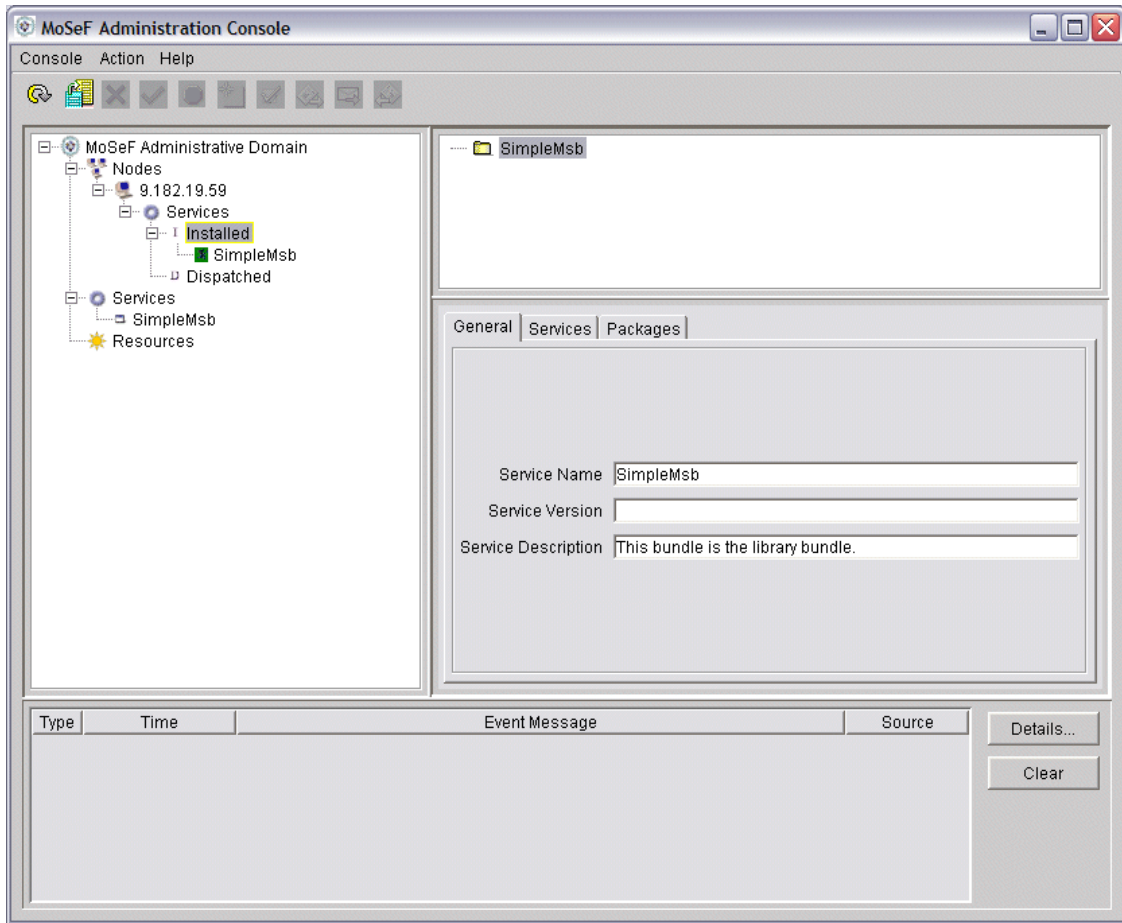
The above case holds good for an active client, i.e. if the client has implemented some fault handler mechanism. In case of passive client like a browser, one might not have any such mechanism. In such cases MoSeF has got a redirecting mechanism of its own, so that whenever a service moves to new location, the new IP address gets mapped to this service URI. After this if some request comes for that MSB service then redirect service of MoSeF redirects the client to new location.

## ***MoSeF Administration Console***

This is the administrative console through which the administrator can monitor all the Hosts which have MoSeF running on it. Certain operations like register, unregister, fetch, retract, dispatch etc can be performed on the MSB. MoSeF instances are administered by passing Remote Messages to the target instance.

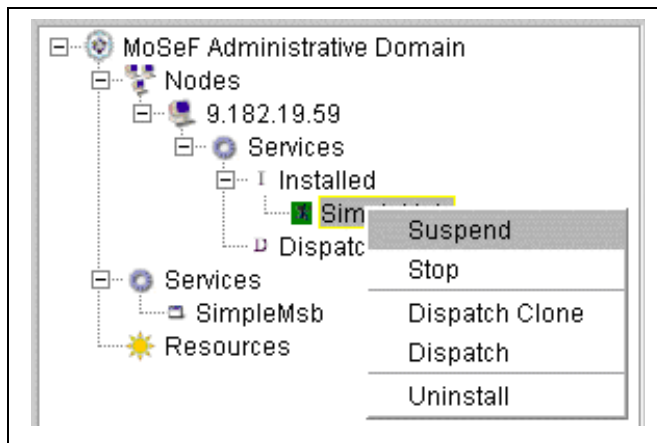
Following are some features of the Admin Console:-

- View of all the Hosts in the domain, running MoSeF, as illustrated below. In the illustration the top node denotes the MoSeF domain. The child to the root node denotes the Hosts that are in this particular domain. By particular domain means the set of MoSeF instances that are registered to a single Hosts Registry to which this instance of the Admin Console is configured to ping for information. In the screen shot below there are two list of MSBs - Active and Dispatched.



- The messages shown in the panel below on the lower portion of Admin Console are the messages being passed between Console and other entities like the Host Registry or the a MoSeF instance.

Besides just viewing the information like various Hosts that are up or list of MSBs that are active on a particular MoSeF instance or the properties of a Host, the administrator can also perform few operations



on MSBs such as Dispatch, Retract and Dispatch Clone. The screen-shot below shows the various available operations. When the Dispatch or Dispatch Clone button is clicked a dialog with a dropdown list will appear which will show all the MoSeF instances that are active. For this list the destination Host can be chosen.

## Mobile Services Framework - Features

The various features supported by MoSeF are briefed in this section categorized as follows: -

- Service Mobility Primitives
- Messaging Primitives
- Data Virtualization
- Service Location Transparency

### *Service Mobility Primitives*

The following operations facilitate the mobility of MSB bundles from one MoSeF node to another. None of the following operations should be performed when the MSB bundle is starting or stopping itself otherwise InvalidMSBOperationException is thrown.

### **Dispatch Operation**

A Mobile Service Bundle can be **dispatched** by MoSeF to a certain chosen location requested by the Administrator / some other MSB / the initiating bundle itself. The Mobility Engine communicates to the MSB when it is going to be dispatched to other location. On receipt of this notification the MSB suspends the ongoing processing and saves its state after which MoSeF packs the MSB with all the state information necessary for it to be properly restored on the destination location. Then it contacts the remote chosen Host and ships the service packet to that location. The MoSeF on the destination Host then installs and restores this MSB.

### **Clone Operation**

A MSB can be cloned and installed locally or can be sent to another location. Clone request can be invoked either by the Administrator or by some other MSB or by the same MSB on itself. Before a MSB is cloned the MSB is informed about it so that it can take do necessary processing before it is clone. If Clone Listener has been attached then its appropriate method is called before cloning takes place.

### **Retract Operation**

A MSB which was previously sent to another location can be brought back through this operation. After its arrival the MoSeF unpacks the jar and installs the bundle in this location.

### **Fetch Operation**

From one MoSeF node, a MSB running at another MoSeF node can be fetched by this operation. After its arrival the MoSeF unpacks the jar and install the bundle in this new location.

### *Messaging Primitives*

Messaging is a key function that facilitates communication across the different components in a MoSeF domain. This section briefly describes messaging under different contexts.

### **Messaging between two MSBs**

There are two modes in which a message can be sent from one MSB to another:

- **Synchronous Mode:-** In this mode the sender is blocked until the message is received by the receiver.
- **Asynchronous Mode:-** In this mode the sender is not blocked. Its execution moves forward after dispatching the message.

However, do note that in both cases, receiver MSB of the message should be running. The MSB which is going to receive a message has to add Message Listeners which can listen to incoming the message, the method of which is called back with the message object as the parameter. One can set the appropriate Message Handlers in initializeMSB() method of the MSB.

Messages between two MSBs transmit via the MoSeF framework. The requesting MSB specifies the MSBServiceId of the destination MSB and the message. MoSeF supports following three types of messages:-

- Text Message.
- Object Message.
- Map Message.

Text, Map and Object messages are used to send String object, map object and Object messages respectively. Incase of Object messaging any Object (basic as well as custom) can be sent, with the only condition that the same object is exported by some bundle at both ends. The corresponding message listeners are provided which can be implemented and there are also adapter classes that provide a 'does nothing' implementation of these interfaces - TextMsgHandlerAdapter, MapMsgHandlerAdapter and ObjectMsgHandlerAdapter for text, map and object messages respectively.

## ***Data Virtualization***

MoSeF has the capability of preserving the MSB's data while the MSB is moving from one place to another.

Every MSB is provided a directory into which it can store its data. Any serializable java object that is provided by Java Platform or by SMF or that has been exported by some bundle in SMF/MoSeF could be stored. MSB can store objects in this directory and this directory will be available to MSB, as it is, even when this MSB has moved to any other location. The data is serialized when the movement of MSB is triggered or it is cloned.

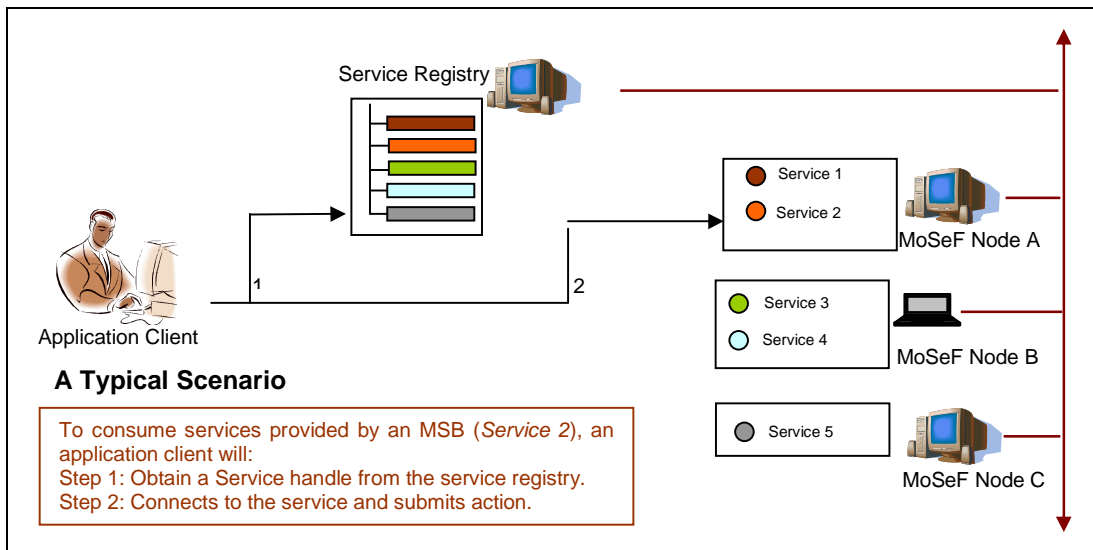
To access the directory the MSB must first obtain a workspace string. Then the MSB can access this directory through the workspace string returned and can create files or directories inside this directory. But once the MSB has moved to another location then again it should do the above to get the location of workspace before accessing the data that was previously put into it. So you have a directory which moves with the MSB.

## ***Service Location Transparency***

Location transparency is a key feature by which client applications are kept transparent of service relocation activities. This has been demonstrated with a sample application packaged as part of this release. This sample consists of two parts: -

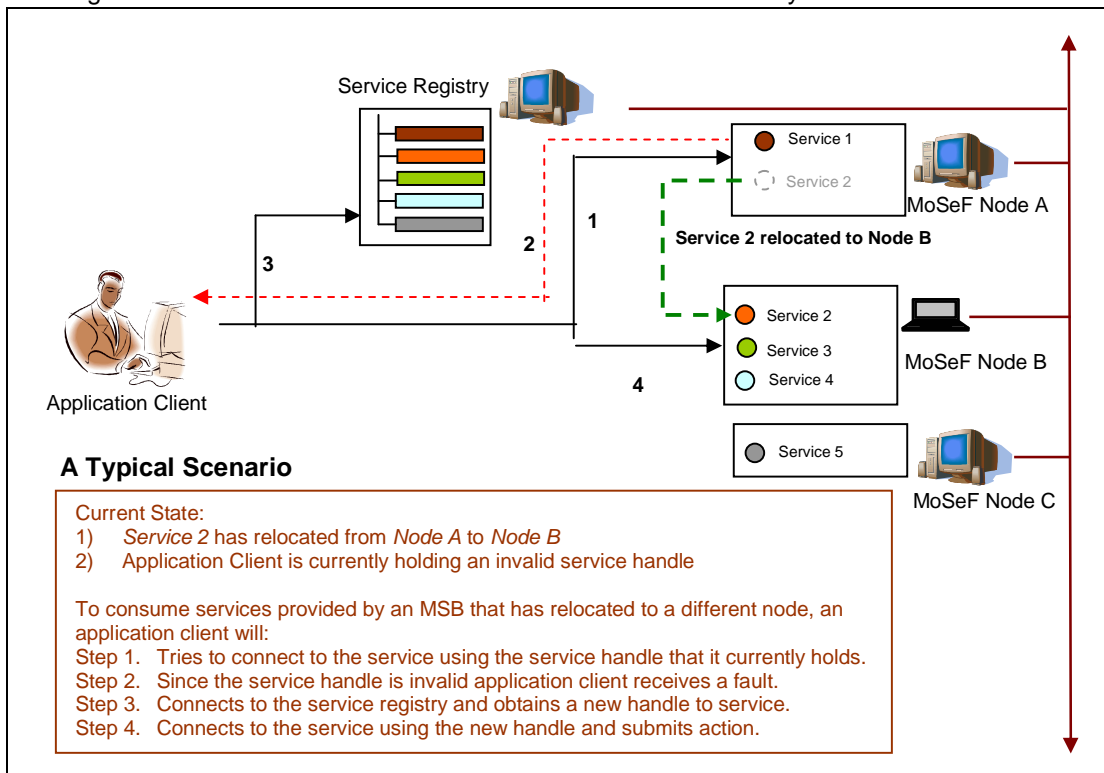
- A Service Provider MSB
- A Client to this MSB

In this Sample, Client accesses the service from the IP obtained from the Service Registry and submits job to the MSB. After the job is completed the Client can obtain the result from the MSB by accessing it from the same URI.



Now if in the meantime the MSB has moved to another location then the Client faults and again accesses the Service Registry to obtain the new location of the MSB. Since the MSB moves with all its state it will still have the results of the job submitted by this client. Now the client accesses the MSB from the new location and obtains the result of the job submitted earlier. All this is done transparent to the user.

At any point of time there can be many instances of client from various Hosts submitting Jobs and obtaining the results. So the services of MSB have all time availability.



## *Stateful Application Relocation*

On Demand Businesses often equip their workforce with multiple devices, so there arise situations where users need to switch across these devices. Most of the contemporary applications running on these devices use techniques such as data synchronization for data transfer. However, with data synchronization techniques it is required that a similar application is pre-installed on the target device to which user is switching. It often leads to restarting and reconfiguration of the application, which breaks the flow and continuity of the user. Rather if the application itself can relocate to a different device along with the state, it shall give a better user experience because a relocated application can retain information such as last edited field, partially filled data form etc.

IBM Workplace Client Technology, Micro Edition (WCTME) provides a SMF based runtime environment for hosting and managing applications. WCTME runs on multiple classes of devices such as Desktops/Laptops, PDAs and Smartphones. And with this effort of enabling SMF to host relocatable services, it is possible to morph the applications written as normal SMF Bundles into Mobile Software Bundles. Thus these applications with some amount of re-engineering will be able to provide consistent user experience even when they relocate across devices.

## Conclusion

We thus conclude that the stateful relocation of service instances, managed autonomously by service hosting environments, will improve the scalability and availability of service end-points. This in turn shall enable enterprise applications to meet QoS promises and Service Level Agreements.

Though SMF was taken as a test bed for MoSeF, the concept is very general and can be similarly extended to various other SOA based hosting environments.

Future releases of MoSeF will envisage the usage of WSRF standard and protocols for component interface definition and components interactions. The MoSeF internal architecture and design will adhere to IBM Autonomic Computing Architecture blueprint.

## Acknowledgements

We are thankful to S Venkatakrisnan for his useful review comments and suggestions towards the betterment of this whitepaper.

## Resources

1. The **IBM Service Management Framework** is an implementation of OSGi Service Platform specification that provides network delivery and management of services.  
<http://www-306.ibm.com/software/wireless/smf>
2. The **Open Services Gateway initiative** (OSGi) website.  
<http://www.OSGi.org>
3. Download the **IBM<sup>1</sup> WebSphere<sup>1</sup> Studio Device Developer IDE** free evaluation.  
<http://www14.software.ibm.com/webapp/download/product.jsp?s=p&id=PDRS-5HMTYH>

---

1. IBM and WebSphere are the IBM trademarks in the U.S. or other countries or both

4. Know more about **IBM Websphere Client Technology, Micro Edition (WCTME)**.  
[http://www-306.ibm.com/software/wireless/wctme\\_fam/](http://www-306.ibm.com/software/wireless/wctme_fam/)
5. You may refer to the related document **Mobility of Service Components - For better reliability and scalability** published on IP.com.  
<https://priorart.ip.com/viewPub.jsp?pubID=IPCOM000010767D>

- 
2. Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
  3. Other company, product and service names may be trademarks or service marks of others.