MSF Technical Report
MSF-TR-ARCH-002-FINAL

MSF R2 Service Architecture

Authors:

Leapstone Systems
Chris Daniel
+1 732 537 6887
chris.daniel@leapstone.com

BT
Mike Bick
+44 207 843 7012
mike.bick@bt.com

Qwest
Ron Egan
+1 303 707 7141
rjegan@quest.com

BT
Ken Mills
+44 117 962 8128
ken.mills@bt.com

Leapstone Systems
Stuart Walker
+44 208 544 8018
swalker@leapstone.com

NTT
Satoru Furukawa
+81 422 59 4227
furukawa.satoru@lab.ntt.co.jp

www.msforum.org
June 2003

ABSTRACT

Through 2002 and early 2003, significant work has been done by the Architecture Working Group to add the details of the application plane to the MSF Functional architecture. The original intent was to release these updates as the Release 2 functional architecture. In Orlando, a contribution was accepted to restructure the architecture in order to make it easier to understand and an activity was begun to this extent. However, the Architecture Working group determined that it was important the MSF publish a white paper for industry use to capture the service architecture work rather than waiting for the complete restructuring. It is the intention of this contribution to provide text that captures the intent of the MSF Service Architecture work for publication as a white paper.

DISCLAIMER

For addition information contact:
Multiservice Switching Forum
39355 California Street, Suite 307, Fremont, CA 94538
(510) 608-5922
(510) 608-5917 (fax)
info@msforum.org
http://www.msforum.org

I. Overview

As communication networks have evolved, one problem that has gotten consistent attention is how to add more flexibility in the creation and delivery of new services. As Internet and IP technologies are increasing service expectations, the downturn in the telecommunications industry has reduced the number of resources that are available to create, deploy, and manage new services. The result is that the expectation for new services is outpacing the ability to deliver them.

This white paper examines the limitations of traditional voice service architectures and proposes a new architectural solution to address these shortcomings. The functionality of service coordination will be introduced and its association with Service Brokering Technology.

II. Limitations of Traditional Service Architecture

In current telecommunications networks the predominant service architecture is the Intelligent Network (IN) approach developed during the 1980's. The goals of the Intelligent Network architecture were:

- Reduce the service time-to-market through decoupling of the service function from the switching function.
- Reduce service cost through deployment of general purpose commercial computing platforms.
- Permit service providers to acquire 'best in class' services from different vendors rather than be limited to/by the offerings of a single vendor.
- Accelerate service development timeframes through the use of graphical Service Creations Environments (SCE).

The Intelligent Network architecture moved the intelligent call processing associated with services from the switches to network based computing platforms. The service handling component of such a platform was termed the SLEE (Service Logic Execution Environment), that executed services defined by the Service Creation Environment (SCE). Service Creation Environments provided a structured environment for the development of services. Services were developed through the graphical manipulation of low level functional building blocks termed SIBBs (Service Independent Building Blocks) or DGNs (Decision Graph Nodes). SCE's could generally support rapid service prototyping and simulation testing prior to network deployment.

At many levels the Intelligent Network architecture was a success. Several IN services were created, deployed and are still running today, together generating billions of dollars in revenue for service providers. The time to deploy new services was reduced from the typical 36 months associated with switch-deployed services to approximately 12 to 18 months.

However, on some levels the Intelligent Network architecture did not fulfill its promises; in particular it did not enable service provides to acquire and/or integrate different services from different vendors. Whilst a standard set of SIBB functions were defined (such as the ITU-T Capability Sets) the implementation of the SIBBs was proprietary and many vendors also extended the set of SIBB functions beyond the standards. This created an exclusive relationship between a vendor SCE and SLEE.

The closed nature of the SCE/SLEE relationship prevented, or at least greatly discouraged, the development of services on any vendors SLEE platform by third party service developers. The result being that all of the services had to be developed by the platform vendor or by the service provider themselves. Since no single vendor or service provider could develop best in class applications across the whole application space, the service providers faced a choice of deploying inferior services or deploying multiple SLEE/SCE platforms (or in some cases Service Nodes which also bundled the switching function into the SLEE). Deployment of multiple service platforms became the norm for most large service providers. This solution resulted in deployment issues around multiple applications interacting with a single call session. The lack of ability to resolve the resulting problems adequately in Intelligent Network architectures, as well as the rise of IP-based service architectures, are two of the key factors that led to the introduction of Service Brokering. While some portions of Service Brokering are often mistaken for an alternate IN Service Creation Environment, in reality these two functions are complimentary. Service creation specifically addresses service development with respect to a specific technology/interface. Comparatively, service brokering addresses how existing services interact in a distributed network.

III. Understanding the Service Vision

While the IN architecture was a good evolutionary step, it is the objective of the MSF to continue to push the service architecture objectives to enable new communication services. In order to access any architectural solution, it is also important to have an understanding of the vision that future service architectures should support. This will allow companies to ensure that technical solutions will also meet the business needs of the future. Here are some of the functions the MSF believes should be included in service architecture requirements.

- Services can be developed by many sources (i.e. vendors, service providers, third party software manufacturers)
- Services can be quickly and easily integrated into the network and back office systems. Operators are looking for a substantial improvement in time to market for new services over the 18 months currently offered.
- Services can be readily integrated to work together and with existing IN services.
- The service elements that can be used across different networks and in conjunction with different base functionality (e.g. fixed and mobile).
- User's services that follow them from device to device and location to location.
- User customization and control of their services
- Multimedia integration
- Multi-domain and multi-carrier/provider integration and interoperation

If there are limitations in a technical solution that will hinder the ability to achieve these objectives, it is important that they are identified early so they changes can be made to support these requirements.

IV. Service Component Architectures

The MSF Architecture assumes from the outset that different services may be developed by different vendors, will likely run on different platforms, and may require different protocol interfaces. At this point in the evolution of communications networks it seems highly unlikely that an all-encompassing standard for service execution will emerge that will enable all services to exist on one platform and persist over time.

The Service Component Architecture, as described in this paper, refers to the integration of multiple, unique service objects acting on a single communication instance without the individual service objects needing to be aware of each other. By allowing these service objects to be developed and maintained independently from each other it is possible for a service provider to mix and match services from different vendors without requiring the vendors to make customized changes in order to get their particular services to interwork. It is the belief of the MSF that IP based services and the introduction of distributed computing technology into the network will drive the rapid adoption of service component architectures.

Component based service architectures have two primary functions: Service Components and Service Brokers.

Service Components are the objects that are bundled or integrated together to create the services that are sold to the customer. There are three types of service components:

1) Application Service Components (ASC): These are the service components that are generally associated with application servers. These components are accessed via signaling and are defined by a set of input and output interaction criteria. For example, in a SIP based service interaction, the specific parameters that will be sent and received in the different SIP messages must be understood. However, the service broker does not need detailed knowledge of how the ASC acts upon those messages. Application Service Components have the flexibility to be defined as either fine grain or coarse grain. An example of a coarse grain service component could be a Class 5 application server or a Universal Messaging Service. An example of a fine grain service component could be Time of Day screening or Short Messaging Service. It is important that service providers have the ability to define both fine grain and coarse grain application service components in order to optimize the signaling and interactions required to provide an end-to-end service.

2) Media Service Components (MSC): These are the service components that are used for interaction with the media path. Media service components are most commonly associated with media servers and include functions like text-to-speech engines, interactive voice recognition systems, and conference/mixing bridges. Some additional media service components that may exist on either gateways or media servers include digit collection devices, announcements servers, and codecs.

3) Data Service Components (DSC): The data service components provide data that are used with or by a service. Intelligent use of subscriber specific data such as presence, location, profiles, directories, and community specifications will allow the creation of value-added services that are specific to the needs of the individual.

Service Brokers are the objects that enable the deployment and integration of the service components. Service Brokers define the rules by which service components exist and interact. The service broker manages the signaling interface to each service component and understands the input and output requirements for interacting with each service component. The service broker only keeps the minimal service state that is required to support the end-to-end service definition and does not need to have in-depth knowledge of the state of each individual service component, though high-level knowledge of the service state (e.g., activated, deactivated) would be required.

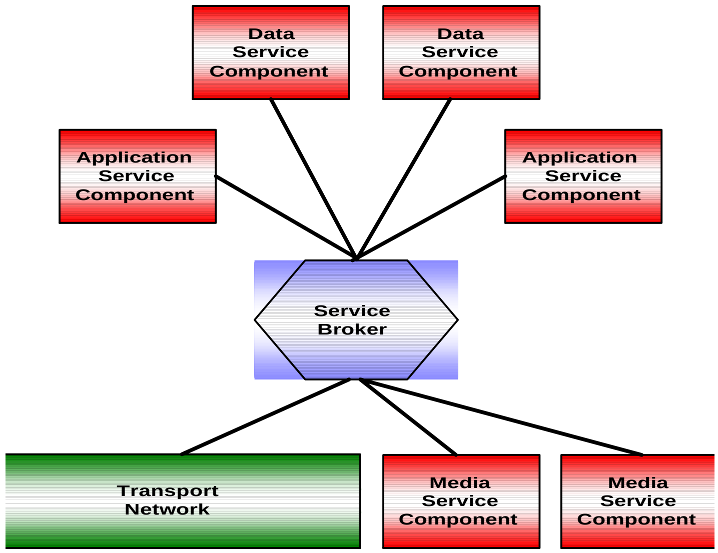

Figure 1 Components of Service Component Architecture

The key attribute of service component architecture is that both the service broker and the service components are intended to be developed as independent functions. Service components can be developed and evolved with little to no impact on other service components or the service broker. The same is true for the service broker in that in can be developed and evolved with little to no impact on the service components other than when significant additional functionality would require interface or behavioral changes to the service components. In these cases, a “release based” approach would be necessary to ensure correct operation.

The amount of dependency between service components, and with the service broker, is sometimes referred to as “coupling”. Tightly-coupled interactions imply that changes in one function will likely require changes in another function. Loosely-coupled interactions imply that changes can be implemented relatively independently between functions. It is the objective of component service architectures to establish loosely-coupled interactions wherever possible by enabling well defined, yet easily extendable interfaces.

V. Industry Adoption of Service Component Concepts.

Standards bodies and other working groups within the industry are defining architectures that include Service Broker-like functions. At this point in time these functions tend not to support the full requirements for brokering described in this paper since the work of these groups has largely been focused on the lower layers of the architecture; however the service and application layers of the architectures are now being aggressively addressed in detail.

The 3GPP Release 4/5 architecture defined by the 3GPP partnership project (www.3gpp.org) includes a Serving Call State Control Function (S-CSCF) which acts as a broker function towards the Multimedia Application Servers (MAS) and legacy (CAMEL) applications. The S-CSCF does include some ‘switch-like’ functions in that it handles call control, resource control and terminal registration.

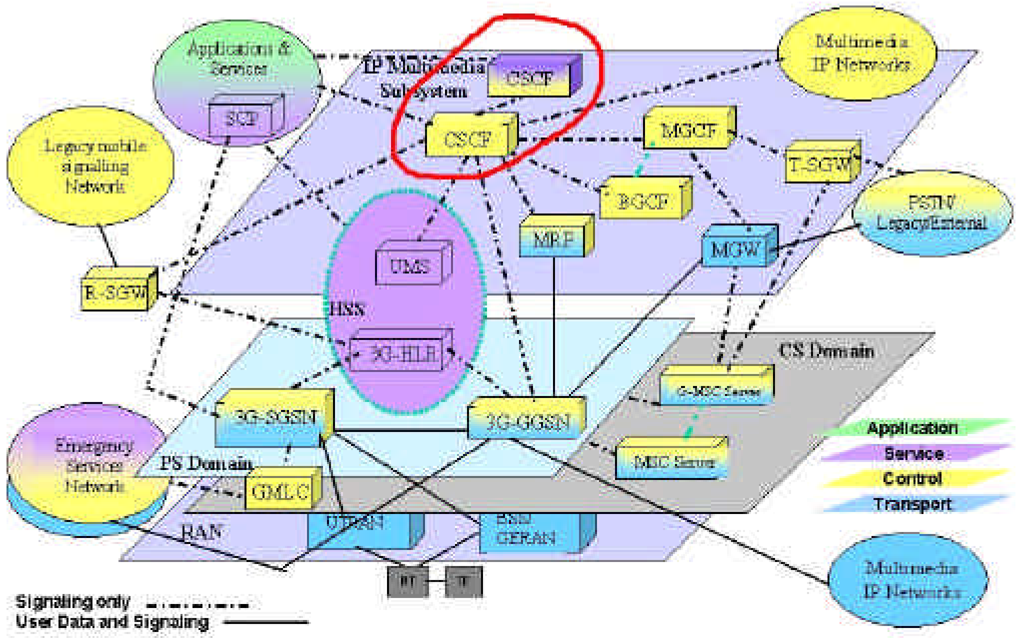

Figure 2: 3GPP Release 4/5 Architecture

The North American 3GPP2 (www.3gpp2.org) were able to take a more 'blank sheet' approach to their architectural definition since there was no requirement for an evolutionary path (from GSM) that 3GPP had to include. The 3GPP2 architecture separates the out the brokering function (Network Capability Gateway) from the call control function (Session Control Manager).

The IETF SIP Working Group (www.ietf.org) has also recognized the role of the service brokering function. In "An Application Server Component Architecture for SIP" (draft-rosenberg-sip-app-components-01.txt), the concept of service coordinator is defined and expanded. Many of the service component architecture concepts in this document were directly influenced by this work and are extensions to further enhance the concepts.

The Multiservice Switching Forum (www. msforum.org) has also specified the need for a service controller in their soon-to-be-released Release 2 Functional Architecture. The MSF functional architecture explicitly recognizes call control, service control (i.e., coordination), service logic, service data, and media services endpoints as distinct functions and defines the relationship these functions have to each other. The MSF is in the process of mapping these functions to common IP service elements (i.e. softswitches, media servers, service brokers, application servers, etc.) in order to demonstrate their use and interoperability in providing end-to-end services as part of the upcoming Global MSF Interoperability 2004 demonstration.

The remainder of this document will focus on the additions that have been made the application plane of the MSF functional architecture to enable a service component architecture. For correlating the generic Service Component Architecture concepts to the MSF Functional architecture, the following high-level rules can be applied:
Application Service Component = Service Logic Function
Data Service Component = Service Information Function
Service Broker = Service Coordination Function
Media Service Component = Logical Media Endpoint Function (Not discussed in this document)

VI. MSF Release 2 Application Plane Architecture.

VI.A. Functions

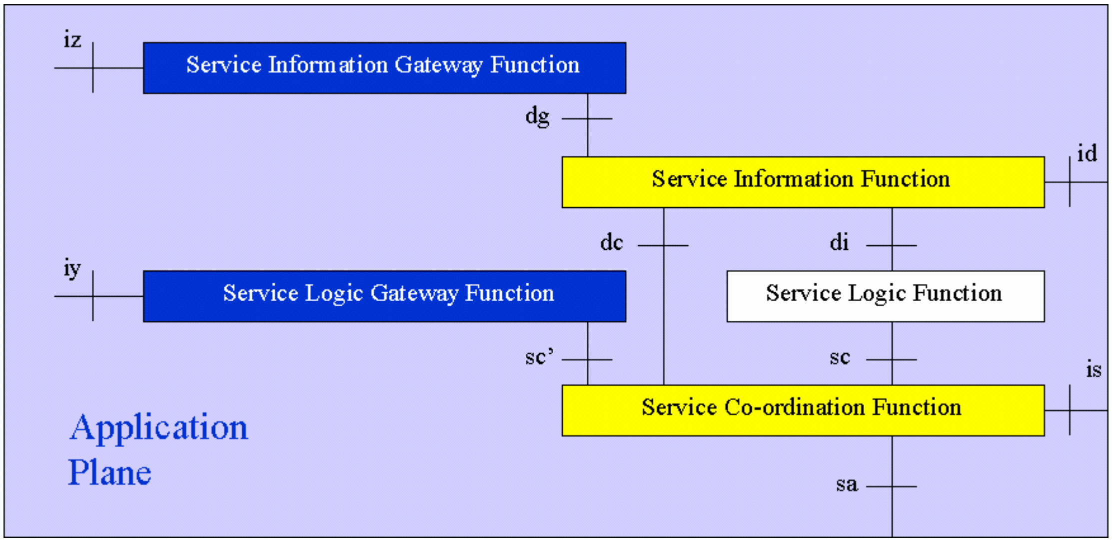

Figure 3: MSF Release 2 Application Plane Architecture

VI.A.1. Service Coordination Function (SCF)

The Service Coordination Function provides functionality that manages for each customer entity (individual and/or group of customers) the integration of multiple services, from multiple service providers, across multiple service domains (which may equate to multiple service provider domains) in a multimedia, multi-services environment. Its major functions are to provide structured interoperation / co-operation of services (SLFs), such that multiple services (SLFs,) can co–exist and will function in an effective, orderly (non-interfering) manner. In order to accomplish this, the SCF coordinates the inter–operability of multiple SLFs for each customer entity. The SCF prioritizes the SLF's precedence for receipt of each "network event message" or other message and manages the state of message distribution to/from SLFs and other system components (SLGF, other SCFs or from the control plane).

This functional component primarily manages the prioritized exchange of messages (commands, notifications, and queries) between Services (SLFs) and the Control Plane. This includes the distribution of messages to and from SLFs and network resources and also between SLFs, based on configured rules, parameters and requests for notifications with specific SLFs by the SCF.

The Service Coordination Function provides a framework that couples individual service logic functions (SLFs) together and with underlying control capabilities through a set of established message interfaces. These message interfaces (protocols) enable the functional modularity that will allow plug 'n' play across vendors for service and control solutions and technology independence from the underlying transport infrastructure. Where a SLF requires an API interface there may be message interface adaptation.

The SCF will communicate with other SCFs which may be in other nodes and/or service provider domains to orchestrate interoperation of services on multiple platforms and in multiple domains. This adds the capability to support various (distributed, peer-to-peer, hierarchical, etc) SCF configurations. Each SCF regulates access to its SLFs from other SCFs.

The SCF may manage the execution lifecycle of SLF instances. The SCF also may manage the SLF configuration lifecycle (introduction, version migration and retirement).

The SCF is implemented independent of the actual “services” delivered by the SLFs. It is strictly a control function and contains only generic coordination functionality. As services are deployed, the SCF becomes aware of the service (SLF) via registration. This registration information is used to determine message distribution requirements and precedence and interdependencies with other services (SLFs).

The SCF mediates the co–operative and intelligent inter–working of multiple SLFs by:

- Registering each SLF based on, at minimum, a classification scheme.

- Routing messages between any of the following: The control plane, SLGFs, SLFs and other SCFs. In routing these messages, the SCF determines precedence ordering and message distribution rules.

The SCF manages the Message Distribution State:

- Receives messages from the control plane, SLFs, SLGFs or other SCFs.
- Applies parameters and policies to rules to determine the service (SLF) priority/precedence and the distribution for each message.
- Distributes each message to the appropriate SLF(s)/SLGFs and manages the distribution state across SLFs.
- Provides coordination of messages from SLFs/SLGFs to other application plane or control plane components to maintain system integrity.

The SCF maintains sufficient information provided by the SLFs/SLGFs and SIFs in order to appropriately perform the above functions. For example, Customer Classification Criteria, Service Provider Policy, Customer Service Preference Order, Current Service State Information, etc. Network level connection or association state (e.g., bearer/media connection topology) is maintained in the Control Plane. Services (SLFs) direct, and/or respond to, changes in the connection (“association”) state via the SCF. Service logic state is maintained within the SLFs not the SCF.

The SCF also performs:

- Real-time correlation of user/customer services to their service profile.
- Policing of messages to ensure In-Contract use and prevention of erroneous or malicious use.
- Policing of the data and/or rules used to establish the SLF/SLGF to SCF and SCF to SCF relationships.

In the computing domain, the SCF is somewhat analogous to a distributed real-time process control system or distributed real-time operating system (OS) where the system/OS manages/controls multiple applications programs/processes and their access to, response to, and control of, underlying, shared physical resources on multiple platforms via a rigorous event and messaging capability.

Examples of SCF to SCF interactions may include:

1. A Hierarchy

   Each individual SCF is part of a group of SCFs with one parent SCF Groups of SCFs may be part of other groups. This may reflect the organizational structure of the "customer"
   e.g.

   Business: Corp/Dept/Group/Individuals
   Family - Personal: Household/Members

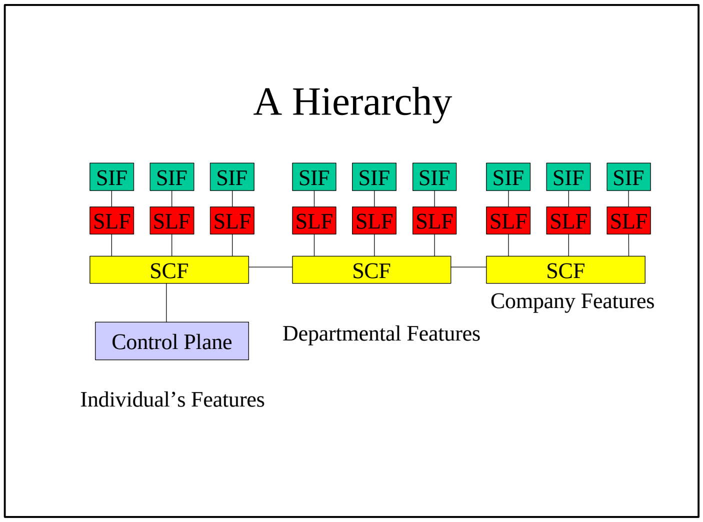

Figure 4 SCF to SCF interactions example 1

2. Master – Slave relationship between two SCFs, inter-domain.
An SCF manages its own SLFs and its relationships to other SCFs across two or more service provider domains. The master SCF enforces trust levels with the slave SCF.

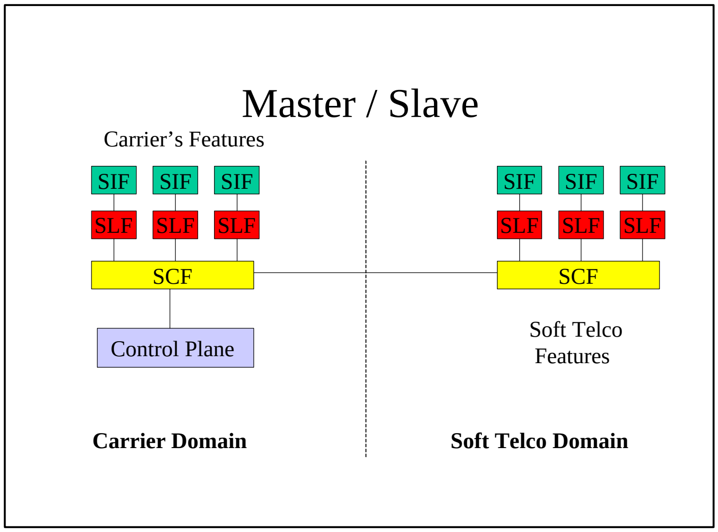

Figure 5 SCF to SCF interactions example 2

3. Peers between two or more SCFs within a Service Provider domain blending of multiple "modes" e.g. Business Mode (via Business SCF) and Personal Mode (via Personal SCF)

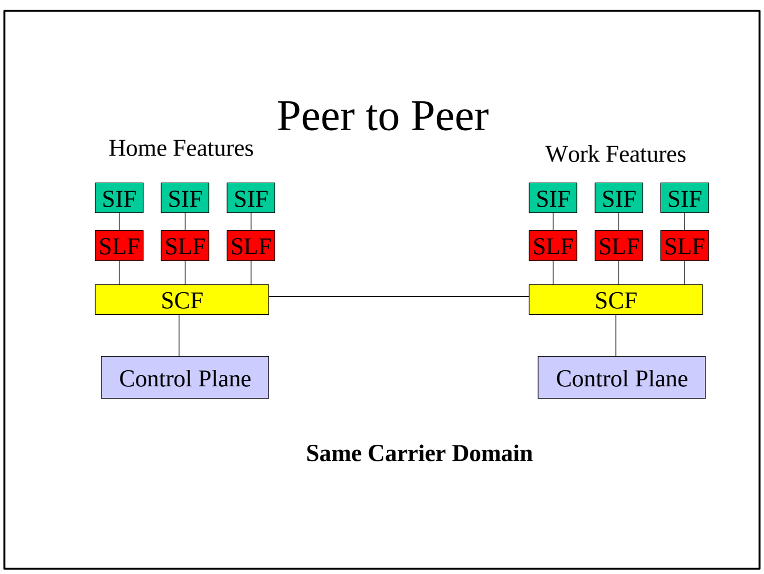

Figure 6 SCF to SCF interactions example 3

VI.A.2. Service Logic Function (SLF)

The Service Logic Function contains the service logic (instruction) functionality that delivers a complete feature to customers as finished goods or products (or components thereof) from the service provider. It may also contain internal system services used by the system to provide converged services capabilities, such as address translation, presence and availability status, or control logic for media processing or content delivery. Customers may range from enterprise/business end-users to residential end-users and from local exchange carriers (LECs) to interexchange carriers (IXCs) to competitive local exchange carriers/data local exchange carriers (CLECs /DLECs) to application service providers (ASPs), et al.

The Service Logic Function (SLF) contains the "Service Logic" (for all system provided services, e.g., voice, data, video, multimedia services) that may direct transport resources or service signaling through control functions via the SCF. SLFs encompass both basic services and enhanced service components.

Services may include both basic services (e.g., a 1FR/1FB two-way "POTS" call, DSL, Instant Messaging, VOIP connection, or private lines, etc.) and value-added or enhanced services (e.g., Call Screening, IM No Solicitation, Voice Messaging, Personal Web pages, URL (Web) Screening, Web Call Centers etc.). The range of service possibilities—including metro–network, ASP, premises, and distributed services—is endless. All service logic functions (SLFs), however, require messaging management functionality in the Service Coordination Function through which messages (commands, notifications, and queries) are exchanged for manipulating underlying capabilities.

Transient information that is specific to the function of an SLF is stored in the Service Logic Function (SLF). Optionally, persistent information which is specific to the function of a SLF may either be stored in the SLF or, if there is a likelihood that the information may need to be accessed by other SLFs in the future, in the Service Information Function (SIF).

Non Call State data (e.g., customer profile, dialing directories, address books), common to multiple SLFs will be stored in or be accessible via the Service Information Function (SIF).

Typically, transient information endures for no longer than the period of a call/session/association. Persistent information has a period of validity that transcends the period of a call/session/association.

SLFs may implement a complete "end-user" service (e.g., basic POTS call processing), a sub-component of an end-user service (e.g., Voice Dialing, Call-Waiting) or an internal system service, or sub-component thereof. An internal system service may, for example, add Security, Management or Class of Service (COS) or Quality of Service (QOS) capabilities (e.g., authentication / authorization, policy) to the end-user service.

An SLF functions independently and operates in an integrated manner with other SLFs/SLGFs under the coordination of a SCF via the sc interface to implement services packages

SLFs requests switching, transport and media resource capabilities (via -sc- interface to the SCF) to perform specific actions.

SLFs have "service logic state" that is internal to the service and which, via the messaging interface though the SCF, effects and responds to changes in the underlying association state.

The Service Logic State Machine:
• Responds to messages from Control Plane, SLGF or other SLFs all via SCF
• Request's network transport and media resource functions via the SCF and Control Plane
• May contain Service Logic State which in part, "mirrors" some or all of the association state (e.g., idle, setup, established, modified, teardown).
• Is driven by messages in conjunction with service–specific data.

SLFs have the capability to expose an abstraction of its service state sufficient for the SCF to manage services (SLFs) interactions.

In the computing domain, the SLF is analogous to a real-time application (or "process control") program, that executes under control of a real-time process control system or operating system.

The execution of Content Delivery and Information Services may be mapped onto the MSF Architecture as SLFs. These services are typically delivered to an end–point in a two–way or N–way association. For Example: Web content servers, Voice Messaging, Video–On–Demand, PSTN information services using IVR platforms, etc.

SLF–to–SLF Inter–working Constraints
SLFs cannot send messages directly to other SLFs, as allowing SLF–to–SLF messaging would remove the SCF from controlling service precedence for each message. Any SLF that needs to message another SLF must therefore do so via the SCF

VI.A.3. Service Information Function (SIF)

The Service Information Function (SIF) stores and provides access to information required for real time execution used by one or more SLFs or SCFs.

The SIF has the following functions:

1. Stores data which is common or shared across multiple SLFs
2. Stores data which is required by the SCF for coordination of SLF processing.
3. May store other persistent data, even if utilized by only a single SLF

4. Provides a consistent, well-defined interface to databases or information stores outside of the scope of the MSF architecture via the SIGF via the dg reference point.
5. Provides access to data held in other SIFs via the id reference point.
6. Controls static and dynamic access rights to data held in, or accessible via, the SIF for MSF elements, which applies to both local and other domains.
7. May control static and dynamic access rights for non MSF elements.
8. May perform information mediation, aggregation, translation required to represent service information as required by SLFs, SCFs or other SIFs.

Examples of persistent data include Dial lists, Equal access information etc. Persistent data excludes for example Call State data.

Where the data is resident outside of an MSF system, it will be accessed through a SLGF.

Typically, the dg, dc, di, and id reference points and SIF functionality is compliant with existing “off the shelf” Database and Information Technology.

VI.A.4. Service Logic Gateway Function (SLGF)

The Service Logic Gateway Function has two roles:

- To allow non-MSF compliant service logic elements, such for example are contained in legacy IN platforms, to operate in conjunction with an MSS, and to interoperate seamlessly with such SLFs as are contained within the MSS.
- To allow an MSF compliant implementation of application plane functionality to control a non-MSF compliant legacy switching system.

To achieve this the SLGF has the following functions:

- It does protocol adaptation between that used by the legacy system and those internal to the MSS
- It provides any functionality required of an SLF not provided by the legacy platform. Examples of this may include such things as registration required for Plug-n-Play

VI.A.5. Service Information Gateway Function (SIGF)

The Service Information Gateway function has two roles:

- To allow an MSS access to external databases in a controlled manner. These may be proprietary, legacy or outside of the telecommunications domain.
- To allow data held within an MSS to be exposed to non-MSF systems.

To achieve this the SIGF has the following functions:

- Protocol adaptation
- Policing of data access

VI.B. Reference points

VI.B.1. The Service Co-ordination reference points (sc/sc') [SCF-SLF/SCF-SLGF]:

These reference points are used to facilitate specific SLF/SLGF processing as part of a coordinated service engagement request. It is also used to pass control instructions from SLFs/SLGFs to SCF (for communication to the control plane (NSICF)) in order to control bearer connectivity, to engage media resources, to redirect signaling and to engage another SLF/SLGF.

VI.B.2. The Data Co-ordination reference point (dc) [SCF-SIF]:

This reference point is used to acquire customer and policy information necessary to coordinate the defined interactions between SLFs as part of a service engagement request from the NSICF. This information contains specific details required to regulate cross domain SLF requests.

VI.B.3. The Service Information reference point (di) [SLF-SIF]:

This reference point is used to acquire and maintain persistent or shared transient service specific information required by a particular SLF to complete its processing.

VI.B.4. The Information Gateway reference point (dg) [SIF-SIGF]:

This reference point is used to acquire and maintain adapted service information that is stored in a non-MSF compliant information source.

VI.B.5. The Data Interworking reference point (id) [SIF-SIF]

This reference point is used to coordinate cross node/domain data access.

VI.B.6. The External Data reference point (iz) [SIGF-external data source]:

This reference point is used to acquire and maintain native service information that is stored in a non-MSF compliant information source.

VI.B.7. The External Logic reference point (iy) [SLGF-external service logic source]:

This reference point is used to facilitate non-MSF compliant service logic processing.

VII. Conclusion

The MSF R2 Application Plane Architecture provides the telecommunications industry with a solid foundation for implementing service component architectures. In accordance with the MSF value proposition, the MSF will leverage the functional architecture to determine appropriate physical architectures to implement the service component concepts. The MSF will then validate the maturity of this functionality by specifying implementation agreements for protocols that are required to support this architecture and will then demonstrate how this architecture should be implemented in interoperability tests.

Introduction of the service architecture over a VoIP network is scheduled as part of the upcoming GMI2004 event. Activities such as determining the services that will be demonstrated, the technologies that will be used to deliver these services (i.e. Native SIP, Parlay API, etc.) and finalizing Implementation Agreements and Test Plans will be pursued prior to GMI2004.

A companion white paper entitled “Service Solutions in Next Generation Networks” will also be published by the MSF to document the known issues that must be addressed by the industry in order to fulfill the vision of the R2 Application Plane. The MSF will work with other industry organizations and within its own value proposition to prioritize and resolve the issues defined in the companion paper to enable the industry to accelerate the commercial availability of solutions that support the MSF R2 Application Plane Architecture.