Ericsson Composition Engine – Next-generation IN

The evolution from circuit-switched intelligent networks toward IMS, the internet, and the web requires an open platform that can seamlessly and efficiently integrate legacy and new services.

The core infrastructure of many of today’s networks has entered a transition phase, as users have begun migrating from classical circuit-switched (CS) networks toward IP Multimedia Subsystem (IMS)-controlled networks. Protecting investments in the intelligent network (IN) service infrastructure of the circuit-switched network and preserving a good user experience for subscribers requires an evolution – not a revolution – toward the new architecture.

IN services are frequently complex and costly to develop and integrate. And given that circuit-switched and IMS networks can be expected to coexist for many years, it makes good sense to reuse IN services within an IMS context where applicable. The services a user is familiar with should migrate with the user toward the new network technology. In this context, existing IN services can be reused as they are and integrated with IMS. Alternatively, existing services can be combined with newly developed components and evolve to true IMS or converged services, thereby profiting from evolved network and terminal capabilities. The approach of choice needs to be motivated on a case-by-case basis.

Also, the ability to enrich existing or evolved IN services by combining them with services from web and internet domains will give rise to even more attractive services with a superior user experience.

Of course, new IMS services should be implemented to serve IMS users as well as the still larger base of circuit-switched subscribers. The Ericsson Composition Engine can help implement advanced IMS services that, in principle, are also accessible to circuit-switched subscribers. Consequently, operators can avoid new investments in legacy technology and services become independent of the access technology, which is a big step toward “communication anywhere.”

Ericsson’s approach to evolving the circuit-switched service layer is to provide an open multiprotocol service platform that can be used for pure circuit-switched services, pure IMS services, and converged services. Furthermore, it provides advanced capabilities for integrating existing circuit-switched services with IMS.

As realized by the Ericsson Composition Engine (ECE), Next Generation IN (NG-IN) offers an efficient evolution strategy that, in many instances, avoids implementing and maintaining multiple variants of the same service within separate domains. Instead, it allows existing IN services to be reused to serve the new IMS user base, making use of a wide range of CAP/INAP versions that implement the service.

A main requirement put on the service layer is the ability to use open technology.
Accordingly, the Ericsson Composition Engine is based on the Open Multimedia Platform framework, which provides a Java EE/JSR289 environment derived from the SUN Glassfish Communication Server.

Operators today are under increasing pressure to differentiate themselves and to provide clear value to their customers, not only in traditional settings but also by embracing new internet services such as Web 2.0 communities. One way of easily achieving differentiation is to combine vertical services, such as telephony, messaging, location, IPTV, push-to-talk, and presence. Likewise, operators can capitalize on the popularity of internet services by integrating them into new types of telecom mashups – realizing typical long-tail applications.

Consider a location-based weather forecast service that is triggered by a push-to-talk voice message: a composite service of this kind could reuse existing telecom enablers, such as push-to-talk, location, and messaging, while adding internet weather forecast and online map services to the mix. The Ericsson Composition Engine enables operators to efficiently implement and deploy telecom mashups of this kind for circuit-switched and IMS users.

**Next-generation IN architecture**

Session initiation protocol (SIP) and IN services differ both in protocol syntax and corresponding protocol state models. Therefore, they are, as part of a user session, invoked in very different ways. SIP services are directly linked into the SIP signaling chain and have full control of the user session. IN services, by contrast, are kept apart from the call session and maintain a control relationship that is limited by the protocol capabilities, mechanisms, and triggers defined for CAP/INAP.

Further, due to the nature of IN standards, business logic is intertwined with protocol-handling logic and protocol-determined state machines. Consequently, use cases that incorporate multiple service technologies require more than the basic mapping of signaling and session state; indeed, any integration between IN and services based on the latest technology also require a significant portion of the business logic that applies to the applications at hand.
The Ericsson Composition Engine provides the basis for Ericsson’s next-generation IN. It is based on an open Java EE server platform for circuit-switched and IMS-triggered services. Advanced composition features make it possible, using a single trigger, to combine locally or remotely deployed services of different technologies in a single composition. The Ericsson Composition Engine is a superset of functions and concepts introduced by 3GPP, including the

- SIP application server (SIP-AS);
- service capability interaction manager (SCIM);
- IM-SSF node service-switching function (IM-SSF);
- service control function (SCF); and
- trigger interaction manager (TRIM).

It can handle the protocols used on the web as well as the service layers of the circuit-switched and packet-switched (PS)/IMS domains.

The aforementioned intertwining of protocol capabilities and service business logic requires a high degree of adaptability from an IM-SSF solution. Flexible alignment and integration of the IM-SSF with services is thus a key feature for converged scenarios. The Ericsson Composition Engine approaches this requirement by providing IM-SSF functionality via the application server platform. With the toolkit available on this platform, Ericsson can tailor IM-SSF functionality to the unique needs of the integration use case at hand. This integration strategy is in clear contrast to a hypothetical solution that favors a generic IM-SSF node as a gateway between circuit-switched and IMS domains. A node of this kind would provide a comprehensive but static set of SSF functionality, thereby failing in practical integration cases.

The Ericsson Composition Engine presents an innovative solution to resolving the feature-interaction issues that arise from the composition of services coming from heterogeneous network domains. Composite services require feature-interaction support and are implemented as a custom mediation function that considers the complete communication needs of the constituent services.

The Java Enterprise Edition (JEE) application server and SIP container (JSR289) form the basis of the solution (Figure 1).

Web services are integrated via the Java API for XML web services (JAX-WS JSR224). The platform integrates CAP/INAP via the Java connector architecture (JCA). IN protocol stacks implement an SSF role that makes it possible to use legacy IN services in the composition logic that resides on the platform. In addition, the IN protocol stacks expose the application server as a service-control function (SCF). Services implemented on the application server can thus be exposed to the circuit-switched domain as IN services.

The protocol stacks take care of low-level protocol operations, such as the coding and decoding of messages and parameters. They also support mediation, but they alone should and do not implement complete service-logic-specific communication procedures and service interaction. The idea is to clearly separate low-level service logic (such as business logic) from support components (such as the protocol stacks that provide control communication capabilities). Nevertheless, in order to provide the best user experience, services must be aware of the capabilities provided by the underlying networks.

Ericsson NGIN services products are typically realized as Java applications in the environment described above. Ericsson and third parties also use this environment for custom service development.

The Ericsson Composition Engine supports interaction with IMS Multimedia Telephony (MMTel). Ordinarily, the IMS core invokes SIP services either before or after MMTel service invocation. The introduction of the Ericsson Composition Engine adds two additional integration points to Ericsson’s MMTel solution. The Ericsson Composition Engine can receive indications from the MMTel service logic and provide information regarding further routing of the call. It can also influence the execution of MMTel services by adding an information element in the IMS service control (ISC) channel. This mechanism enables ECE-based service composition with, for example, the included MMTel-terminating service but without MMTel call forwarding.

Advanced service composition

The Advanced Composition Engine is a function in the Ericsson Composition Engine that makes it possible to handle feature interaction between services that run on different platforms. Equally important, it creates and executes workflows by instantiating the abstract descriptions of the composite service (Figure 2).

Service-composition technology, as provided by the Advanced Composition Engine (ACE), targets integrators, enabling them to quickly adapt or real-
ize new adapted business offerings from existing services. Advanced composition follows SOA principles, and essentially extends the SOA tooling to the IMS and SS7 worlds.

The Advanced Composition Engine facilitates the rapid and dynamic implementation of customized mediators without having to write, generate, or compile classical source code. This approach to service composition flexibly reuses existing service components and combines them with new functionality. Most mediation tasks can be covered by pre-designed services (for example, translating user addresses between SIP and circuit-switched addressing schemes) from an existing component toolbox.

The Advanced Composition Engine is technology-agnostic and allows components from different platforms and technologies to be mixed within a single composite service. It currently supports SIP services, CAP/NAP-based IN services, SOAP/web services, and JSON RESTful services. Further technologies can be added efficiently without changing the core composition concepts or the way in which composite services are designed.

The Advanced Composition Engine helps developers to reuse functionality – they need only define the functions to be included in the composite service, and related dependencies and constraints.

The Advanced Composition Engine selects, in run-time, the constituent services to be executed and maintains the overall composition session context. Constituent services are selected by matching developer-stipulated requirements and constraints with the current system state and the service descriptions stored in a service repository.

The accumulated state in a composite service enables the management of service interaction between constituent services across technological borders. Rather than being based on protocol abstraction, the full network triggering context, including network addressing information, is available in the composition context. Any protocol parameter can be used as input for decisions or as input for services used in the composition. Mediation functionality can be added to the composition through components that use the shared state – for example, a component that translates the user address received in an INAP message into a SIP user ID.

To summarize, advanced service composition:

- enables applications to react flexibly and to tolerate changes in the environment – such as different deployment, multi-tenancy, system load, and system faults;
- dynamically selects the services to be included in the composition at the time of execution. This gives a more flexible reaction to run-time conditions than a solution that is based on pre-defined workflows;
- provides a toolset for managing interaction between services that originate in different technological worlds; and
- enables efficient adaptation to the requirements of new use cases while reusing large parts of previous work.

**Graphical creation environment for composite services**

Advanced service composition is carried out through a graphical development environment that integrates

- a graphical creation tool for modeling composite services;
- a monitoring and debugging environment for composite services;
- a front-end to the service repository; and
- an environment for deploying and managing the compositions on the Advanced Composition Engine.

**Next-generation IN use case**

The following example, which originates from a recent customer proof of concept, demonstrates an IMS-triggered SIP/SS7 composite service involving on-line charging and VPN control implemented in the circuit-switched network.

The limited ability of the core network setup to address number normalization and to filter emergency numbers as required for the use case at hand causes an exceptional implementation. In this respect, the example shows how easily the Ericsson Composition Engine can address integration-specific requirements.

It is important to note that the integration of a VPN service with IMS and online charging does not need to be realized as described in this example. In particular, Ericsson’s VPN service is already integrated with online charging, and to fully profit from network and terminal evolution, it will evolve to be integrated.

**BOX C**

**SOAP, JSON, RESTful**

SOAP is short for simple object access protocol. JSON is short for JavaScript object notation. RESTful is short for representational state transfer.

**BOX D**

**S-CSCF and IFC**

S-CSCF is short for serving call session control function. IFC is short for initial filter criteria.

**FIGURE 3** Example composition skeletons (originating and terminating) for realization of the use case.

![Example composition skeletons](image-url)
with IMS directly. However, this customer example demonstrates the ability of advanced service composition to seamlessly integrate circuit-switched/IN and packet-switched/IMS services within a single application.

Example: Subscriber A, who has an assigned VPN group (for example, a short-numbers group for family members), calls Subscriber B, who also has an assigned VPN group. When Subscriber A calls an emergency response number (such as 112 and 911), the Ericsson Composition Engine only invokes the number normalization service; that is, it skips the VPN and online charging service, and forwards the SIP INVITE to the core network immediately after the number normalization sends back the SIP INVITE. The number normalization service handles special numbers like emergency numbers by inserting a location suffix in the B-number in order to redirect the call to the correct emergency center.

Figure 3 shows the composition skeleton for the above example. Using either of these services, the composite service handles user interaction, releases the call, or both. The services can perform user interaction without involving the composition. In the SIP paradigm, releasing the call implies that the next service is not invoked.

In this example use case, there is no need to act on subsequent events (SIP answers) in the composition, because there is no need to link new SIP services or web services based on the occurrence of these events. The skeletons are only executed at SIP INVITE.

The two composition skeletons in Figure 4 are for the originating side (left) and the terminating side (right). A check is performed on the originating side to determine if an emergency number has been called. This check determines whether or not the VPN and online charging service should be triggered or bypassed. The VPN is always used in the terminating case. The constraint in the top element of the skeletons represents the selection of composition based on information stored in the S-CSCF/IFC and passed on in the SIP AS uniform resource identifier.

Figure 4 shows the high-level architecture for SIP-triggered compositions. The SIP execution agent (EA) integrates the Advanced Composition Engine with the SIP container and SIP services.

The online charging integration service is a locally deployed SIP service that integrates with online charging via Diameter. User interaction may be a factor for both VPN and online charging integration services. However, only the VPN user interaction is visualized in Figure 5. An establish temporary connection (ETC) message from the VPN is mapped by the INAP EA into setting up a SIP dialogue using the E164 address in the ETC. The INVITE is routed by the IMS network via a circuit-switched/IMS gateway to the specialized resource function (SRF), which contacts the VPN via an assist request instructions (ARI) message. An alternative approach is to chan-
Figure 5 also shows the difference between the SIP and CAP/INAP paradigms: SIP services are added to the SIP signaling chain, whereas IN services maintain a control relationship with the actual call session in the network.

Summary and outlook
Ericsson NG-IN is a composition platform that can handle the various protocols used in the web as well as the service layers of the circuit-switched and packet-switched/IMS domains. In addition to supporting migration use cases between circuit-switched/IN and packet-switched/IMS, the concepts presented in this article allow operators to realize new IN and IMS services and to enrich existing IN services with services from web and internet domains.

The Ericsson Composition Engine (ECE) is an innovative solution to abstracting and resolving the feature-interaction issues that result from the composition of services coming from heterogeneous network domains, as part of the IN evolution toward IMS and internet services. Further, it enables operator-specific service orchestration.

The architecture of the Ericsson Composition Engine enables IM-SSF, reversed IM-SSF, SCIM and TRIM use cases without being restricted to a pure brokering node or to compositions based exclusively on call-control protocols.

Existing services can be flexibly reused and combined within new contexts. This is supported by a service composition framework that enables mediation between service components across technological borders.

References and trademarks

1. Introduction to IMS. Ericsson white paper, March 2007
3. MMTEL – a standard for multimedia services over IMS. Ericsson white paper, July 2008
7. Rogier Noldus et al.: IMS multi-access, single core, consistent service delivery. ICIN 2008

Java is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries.