Ericsson Technology Review

Charting the future of innovation

#3, March 2024

Evolving service management toward intent-driven autonomous networks

Ericsson Technology Review — Charting the future of innovation | #3, March 2024

Evolving service management toward intent-driven autonomous networks

Authors:
Jörg Niemöller, Elisabeth Müller, Massimiliano Maggiari, Kamal Maghsoudlou

Intent-based autonomous networks entail new capabilities and interaction schemes that impact the use, exposure and management of services. Our goal is to help communication service providers seize the opportunities that service management evolution presents without the risk of disrupting earlier investments and critical business processes.

ISSN 0014-0171 284 23-3406 | Uen

Ericsson, SE-164 83 Stockholm, Sweden
Phone: +46 10 719 0000

The degree of autonomy in a network is measured in terms of the amount of human action that is required to make operational decisions and implement them in the network, with the zero-touch paradigm as the ultimate goal.

The path toward an autonomous network is achieved through the implementation of network operation that is data-driven, self-aware, self-healing, self-optimizing, conflict-resolving, continuously learning, self-adaptive and proactive. An autonomous network is characterized by its ability to:

- Continuously monitor all relevant metrics
- Reason about its own state and identify issues and opportunities to improve
- Automatically find and apply solutions
- Continuously seek to optimize its state
- Detect and mitigate conflicting requirements
- Learn from observation
- Adapt to new situations and demands
- Predict, anticipate and avoid problems.

Ideally, human interaction with an autonomous network should be limited to telling the network what it is expected to achieve and then retreating to monitor that the network is operating as intended. Every situation that requires further human intervention reduces the level of autonomy. These situations are not only the result of errors but can also be caused by the introduction of new products and services, or by changes to the network, such as upgrading or expanding resources. A highly capable autonomous network would have the ability to adapt itself in these cases.

The concept of intent

The expression and management of operational requirements as separate information objects is an essential step toward autonomous networks. This approach enables a higher level of abstraction as well as separating knowledge about the requirements from the implementation of requirement handling provided by service orchestration and assurance solutions. Until now, the requirements on service characteristics and behavior established and assured in service management have been specified in Service Level Objectives (SLOs). SLOs are life-cycle-managed as artifacts in the service catalog. However, the way SLOs are used in typical implementations of service management with orchestration and assurance is relatively static. SLOs are usually human-determined and human-managed artifacts. Furthermore, changes to the requirements expressed by SLOs do not propagate to existing service and resource instances. They are considered at service creation and configuration time only.

Intent is the next evolution of requirement-bearing knowledge objects [1]. Intent has a dynamic life cycle based on application programming interfaces (APIs) that are specifically designed to allow automated systems to dynamically set and modify requirements as needed at

Terms and abbreviations

API — Application Programming Interface | **CFS** — Customer-Facing Service | **E2E** — End-to-End | **LCM** — Life-Cycle Management
RFS — Resource-Facing Service | **SLO** — Service Level Objective

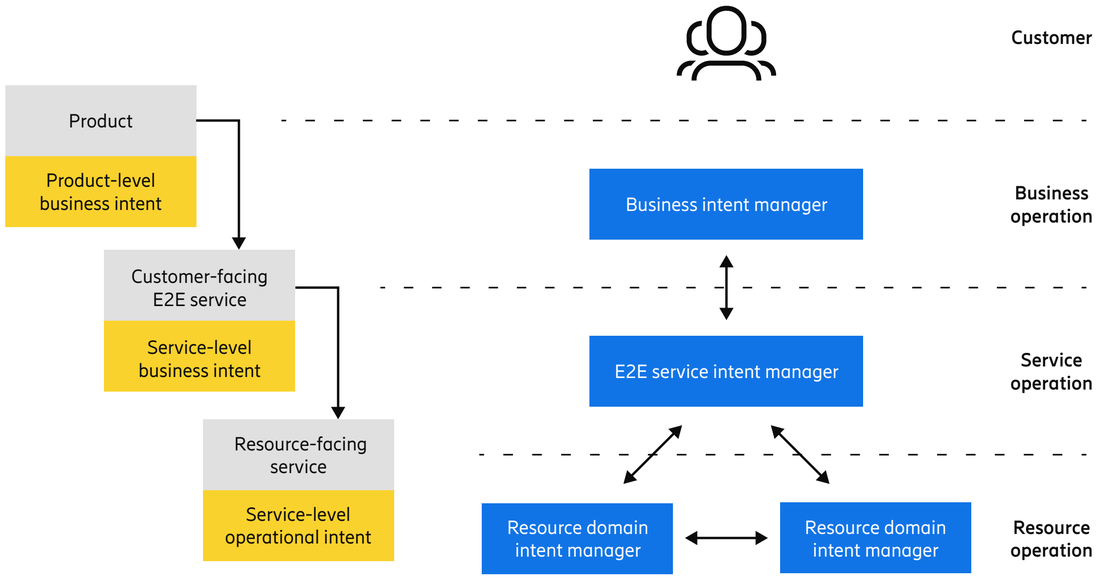

Figure 1: Product and service breakdown in the layered intent management architecture

any time in the life cycle of associated services and derived service instances. A change of requirements expressed by intent is expected to take immediate effect also for compliance of already deployed service and resource instances. This effect on instance level is a key difference between typical SLO and intent-based systems. An SLO bears requirements associated to services, while intent also affects service instances with the expectation that service instances adapt to intent changes. Designing service orchestration and assurance accordingly is a central challenge in the transition of service management toward intent-based operation.

Intent APIs defined by various standard-defining organizations such as TM Forum and 3GPP (3rd Generation Partnership Project) specify methods to check feasibility, negotiate requirements and provide guarantees. Intent-based operation is also embedded in the generic architecture of autonomous networks, in which automated intent managers communicate with each other regarding requirements using intent and provide feedback on compliance to requirements using intent reports.

It is important to note that intent is not necessarily tied to specific services. By providing general requirements for a network domain, intent can indirectly impact how services and products are handled by introducing additional requirements with which they need to comply. However, the recent addition of intent into dedicated service management APIs is the key enabler for the evolution of core commerce processes, and service orchestration and assurance implementations toward intent-driven systems. Intent would complement or replace SLOs as the requirement-bearing artifact. Therefore, the implementation of service management solutions needs to evolve to support the dynamic aspects of intent requirements.

Autonomous network architecture

An autonomous network is subdivided into autonomous domains according to the generic architecture proposed by TM Forum [2]. These autonomous domains are allocated within layers of the network operation software stack, residing on the business, service or resource layer, for example. Within a layer, multiple autonomous domains may exist in parallel or in further subordinate layers.

According to this generic architecture, autonomous domains have a specific responsibility scope within the overall autonomous network. Autonomous domains also interact by setting requirements using intent. This is a common pattern: an autonomous domain has terminal goals that it is required to meet and optimize. To reach its goals, it also needs other domains to contribute to and comply with certain requirements. Intent is used to communicate and manage these requirements in the various layers in our layered approach to intent management [3].

Every autonomous domain within an autonomous network implements intent management and contains an intent manager, which can be realized using a cognitive loop [4]. Intent managers are the entities that communicate with each other using intent to set requirements. An intent manager also coordinates with other functions within its domain to determine how to achieve the requirements.

Intent managers within autonomous domains are the key entities involved in intent-based operation. Every intent has exactly one owner and one handler. An intent manager plays the role of intent owner if it defines requirements with which another domain needs to comply. If, on the other hand, it receives requirements that it must comply with in its own domain, it assumes the role of intent handler. The right side of **Figure 1** shows the allocation of intent managers in the business, service and resource operation layers.

Intent API evolution

Intent managers use APIs to manage requirements expressed in intents. TM Forum has published the TMF921 API for general purpose intent management. It specifies the communication between intent owners and handlers, enabling operations such as providing intents, reporting on compliance status, modifying requirements and removing intents. It also defines advanced operations such as requesting proposals, providing feedback or checking the feasibility of requirements.

TM Forum is currently using Ericsson's contribution of extensive research and development work in the area

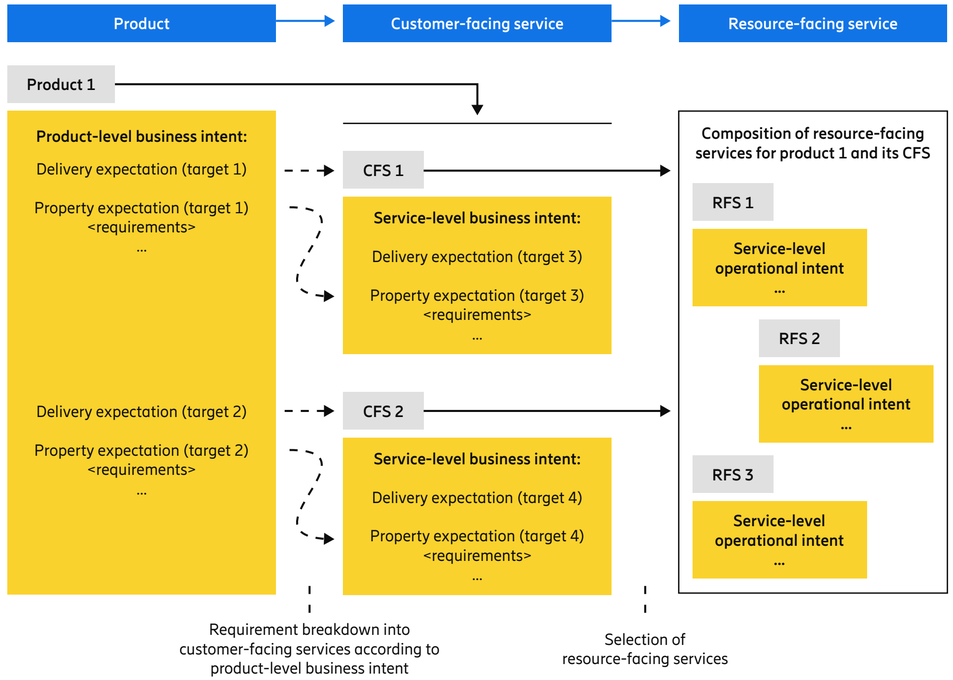

Figure 2: Service referencing from within an intent with example

of autonomous network technologies as the basis for integrating intent management capabilities into its existing APIs for product and service management, as well as its Open Digital Architecture. The API integration process involves several key changes. The first is the addition of intent specifications to the service catalog with links to product and service specifications. The related APIs are TMF620 product catalog management and the upcoming version of service and resource catalog management TMF633 and TMF634.

The second key change is the capture of intent values or references during the ordering phase. These intents are business intents when associated with ordering a product and its associated customer-facing services (CFSs). They will be converted into operational intents when ordering the services used to realize products. Intents are communicated within the product order using TMF622 or service order using the TMF641 API. These APIs will also allow the life-cycle management (LCM) of intent, including dynamic modification and removal.

The third key change is the creation of a new operation in intent-based systems: intent reporting that creates an implicit feedback channel about the success of handling requirements. Creating reporting expectations will be part of the TMF622 and TMF641 ordering APIs. The intent handler will be responsible for assembling the intent report in line with the reporting expectations and making this information available for external consumption by the intent owner and/or others. The addition of intent qualification to the TMF679 API product offering and the TMF645 API service qualification is another important change, which makes it possible to check the validity of intent against the intent specification. Lastly, changes will be made to make it possible to maintain intents in the product and service registry at the instance level, in relation to the respective product or service instance.

The introduction of intent directly into service management APIs is important for a seamless transition of current business processes and a gradual introduction of intent. Further intent management capabilities will be introduced at a later stage, which will make it possible to explore the feasibility of compliance, negotiate requirements, and both ask for and provide guarantees. This can be realized as part of qualification, service order or through the dedicated TMF921 intent API.

Service management with intent

The service management domain is the starting point for the introduction of intents, as it is the heart of network operation, and intents are a prerequisite of autonomous operation.

The left side of Figure 1 allocates products and services and their respective intents within the layers of an autonomous network. The business intent manager is realized by the core commerce function. Business intents will be captured in the core commerce function during product ordering. When the product order is decomposed into service orders, the business intents must also be decomposed and transformed to service intents on multiple levels. The first decomposition step must be applied to the CFS level: these are the services the customer is aware of as part of the ordered product. This is also the level where the classic requirements (phrased as SLOs) would be attached.

Figure 2 provides an example of how the information provided within the intent is used in the successive decomposition process. The end-to-end (E2E) service intent manager will utilize the delivery expectation mechanism. A delivery expectation specifies that something needs to be delivered and provides details relating to it. The intent of a product would specify the set of CFSs to be delivered.

DEFINITION OF KEY TERMS

An **intent manager** is a function within an autonomous domain that receives requirements expressed by intent through APIs and coordinates with other functions within its domain to achieve the requirements.

An **intent owner** is an intent manager that defines requirements and thus is the source of an intent.

An **intent handler** is an intent manager that receives an intent and is thus obliged to comply to the requirements within its autonomous domain.

An **intent specification** is a template containing rules and constraints for valid intent content associated with a product or service.

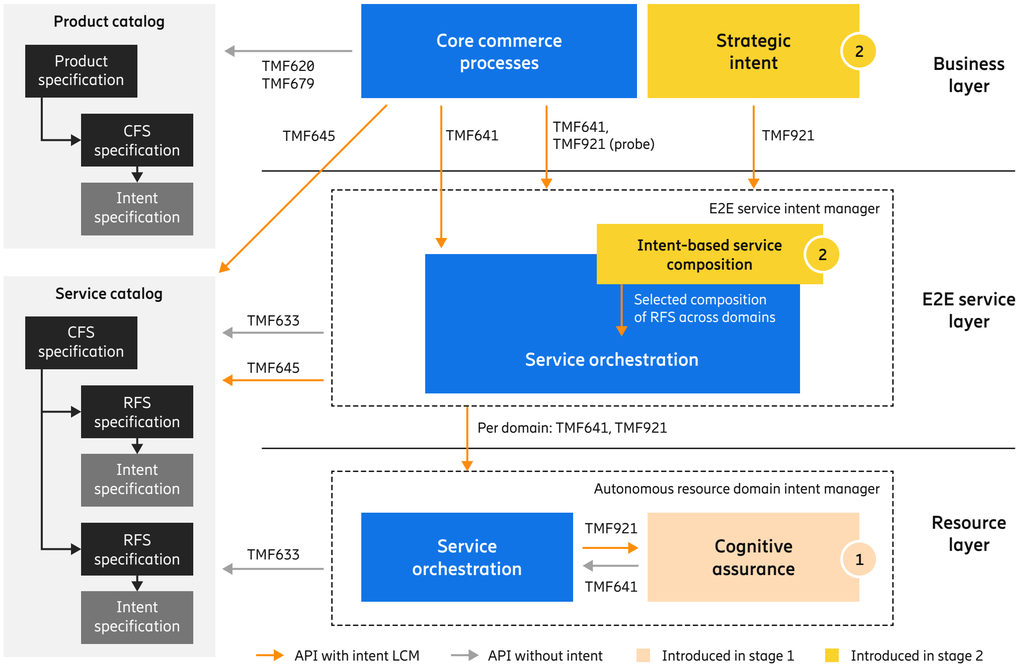

Figure 3: Staged evolution of intent-aware service management

Further requirements stated in the product-level intent, for example, using property expectations and key performance indicator-based conditions, are distributed into the intents for the respective CFSs.

The decomposition of CFSs into applicable resource-facing services (RFSs) with associated operational intents is the responsibility of the E2E service intent manager located in the service operation layer. This intent manager involves functions such as service orchestration.

RFSs are not exposed to the customer. This means the E2E service intent manager has more freedom to compose suitable services. The mechanism of delivery expectations within intents is flexible and can span from specifying a concrete service to be used to loosely categorizing the kind of functionality needed without specifying which concrete service to use. Intent expectations in the CFS would be used to describe properties of suitable RFS choices.

For example, the service-level business intent associated with a CFS may state that a communication service is needed, providing an E2E latency of 100ms. The E2E service-level intent manager may choose a network-slice-based resource-level service and specify that the radio access network must achieve a latency of 60ms. Consequently, the intent manager selects an RFS matching the needed functionality and breaks down the E2E requirement into a domain-specific contribution.

The receivers of the RFS orders with intent are the resource-domain intent managers. Intents associated with RFSs are referred to as operational intents rather than business intents. Intent managers at this level cater for further translation resulting in domain-specific actions. Examples of this could be provisioning and configuration actions toward network resources or the sending of further intents if resource and network functions are also intent-aware.

Evolutionary introduction of intents into service management

Rather than being disruptive, the introduction of intents into service management should be a smooth evolution in multiple stages. Communication service providers must be able to adjust and upgrade their orchestration and assurance stack without interference to business continuity.

Figure 3 illustrates our proposal for a staged evolution of intent-aware service management. The starting point and base for the evolution is stage 0: SLO-based orchestration and assurance. In this stage, requirements on service characteristics are already separated from service order invocation and captured with SLOs. The service orchestrator uses service-specific policies to analyze and decide how to provision and configure the network considering the SLO-defined requirements. Artifacts such as TOSCA (Topology and Orchestration Specification for Cloud Applications) models and Business Process Modeling Notation-defined processes guide the orchestrator's work. The orchestrator also determines the assurance processes needed for automated monitoring and healing.

The introduction of intents should be a smooth evolution in multiple stages.

In stage 0, neither intents nor dynamic LCM of intent requirements involving the possibility to dynamically change the intent content at any point of the service-instance life cycle are used. This is to preserve compatibility with current product and service management processes. Over time, product and service management can evolve in multiple stages to introduce these concepts.

Stage 1: intent-based assurance

Stage 1, shown in the lower part of Figure 3, follows the optional, intent-related extensions to product and service management introduced by TM Forum. Intent is introduced in service operation and replaces SLOs for services using intent-based operation. The business intent details are captured by core commerce processes together with CFS orders utilizing extensions to the TMF641 service-ordering API. This API will make it possible to order services and also manage the respective intent. It is possible but not

mandatory to start capturing business intents when ordering the product. This is already supported by the latest version of the TMF622 product order management API.

Intent specifications are introduced in the product and service catalog and linked to product and service specifications. Intent qualification can be done together with product or service qualification against the intent specification using the TMF679 or TMF645 APIs. Further intent feasibility check and requirement negotiation operations can be performed using the TMF921 intent API or as part of the qualification using TMF645. The intent instance is persisted with the E2E service, and the life cycle of the intent requirements must be managed together with the service.

The E2E service orchestrator would apply a catalog-driven composition to determine the RFSs to be configured and instantiated and would use the intent-defined requirements to make decisions in provisioning. A significant change in stage 1 would be the introduction of an intent-aware assurance loop. The orchestrator will use intents to arm the assurance loop with the requirements to assure using the TMF921 API. These can be the same requirements received in the order or a variation of them based on additional service- and resource-specific concerns to be assured. The related intent reporting will generate notifications to Service Level Agreement management related to the service quality and requirement compliance.

The main task of the intent-based cognitive assurance loop would be to monitor the operated domain and keep it compliant to the operational intent requirements. It would observe through a measurement and analytics infrastructure and act, for example, by reconfiguring resources through network management or through the orchestrator by modifying services or ordering additional ones. A self-adaptive, intent-driven control loop [4] is well suited for this purpose.

In this stage, advanced features such as decision by utility maximization can be introduced. This concept uses intent to communicate utility information. Utility is a concept specified within intent, which is realized by a utility function that maps an observed metric to a normalized score. This score expresses how valuable the observed value of a metric is. Utility-based decision refers to selecting the most valuable solution alternative with respect to resource availability and currently valid requirements. As such, it also addresses conflict resolution.

As all artifacts, including intent, are optional, the introduction of stage 1 and transition toward intent can be done gradually and on a per use-case and feature basis. For example, intent may be used for new use cases and those services that would profit from dynamic requirement management, while established services continue to be based on SLOs.

Stage 2: intent-based composition and service selection

After intent-based assurance is established, the next stage is the allocation of an intent-handling function as a layer inside the intent manager. Figure 3 shows the allocation of intent-handling functions as a separate process (the selection/composition of resource-facing services), but in reality, they will be implemented as part of the orchestrator. As a result, the service order with the intent will terminate in the intent management layer. Further decisions are entirely based on the fulfillment of the stated intent requirements.

If the intent requirements for the new intent are not met, the loop will pick up on it and propose corrective actions. One typical solution in this case can involve the orchestrator and lead to provisioning of resources according to the ordered service. As part of this solution, the intent manager can select the set of RFSs to be used and determine the intent requirement breakdown per subsequent resource domain. This approach can, however, result in no initial action if all requirements expressed by the new intent are already met. The intent manager would then immediately start monitoring continued compliance and would not act until the requirements are no longer being met.

Transition toward intent can be done gradually and on a per use-case and feature basis.

In this embodiment of a service layer intent manager, all operational decisions, including initial fulfillment and subsequent assurance, healing and optimization, are counteractions for fixing the situation that the network is not meeting all its requirements in the most optimal way. These actions can involve the configuration of resources, selection and orchestration of RFSs or the breakdown of requirements received in a CFS intent into resource-domain-specific intent. This means, the intent-based service composition provides a flexibility layer based on the dynamic composition of services, with continuous monitoring and optimization.

This concept also introduces a strong separation of concerns and decoupling of system layers, as the resources used to implement a customer requirement and deliver the services a customer expects would not be directly implied by the services or products being ordered by the customer. Instead, the autonomous service layer would find solutions based on the available RFSs to optimally utilize resources. This also means that the service management system would not be bound to inflexible vertical integration of predefined and coupled artifacts. This is an environment in which methods based on artificial intelligence, with its potential to find creative solutions and optimize beyond human capabilities, can deliver great value.

Stage 3: adapting existing resource-facing service instances

The intent-based service composition function introduced in stage 2 uses RFSs flexibly, but still chooses from a catalog of predefined services. In stage 3, the idea is that the service instances themselves can evolve. The autonomous network could introduce new service management artifacts or variate existing ones to better represent available resources. Furthermore, changes to the resource usage and configuration can be implemented through modification of service instances. At this stage, the autonomous network has full adaptation capabilities and can reshape existing service instances in line with new requirements involving new service specifications.

For example, new resources may be supplied by a new business partner. They may also provide better scaling capabilities or better response times or be more efficient from a total cost of ownership point of view. These new resources will be made available to the orchestration function by adding new resource specifications and

associated RFS specifications. New service orders can utilize them immediately, but it would also be beneficial to replace already provisioned resources and service instances.

In stage 3 of the evolution, an autonomous network can decide to reshape existing service instances and eventually transition them to the new resources, once they are made available. The decision to transition service instances to new resources would be an optimization decision taken by the intent-handling function, which is continuously producing new solution alternatives to improve the system state and raise business value. This process would assess that a dynamic re-composition action will lead to a higher level of intent fulfilment and business utility. In other words, once the new resources are available, the intent-management function will identify the optimization potential and execute the required actions to achieve it. This involves the recomposition of the service by the service management system. This type of dynamic self-adaptation is a key capability for autonomous networks.

Conclusion

While it is clear that the creation of highly capable, zero-touch autonomous networks will require major changes in network operation and service management, our research demonstrates that it is possible to introduce the necessary changes gradually with increasing levels of autonomy over time. Self-adaptive capabilities are often challenging to achieve and require the use of artificial intelligence methods in combination with interfaces and architectures that strengthen abstraction and separation of concerns between the layers of network operation. Service management provides an excellent abstraction of resources for this purpose, especially if combined with dynamic requirement management of intent-based interactions.

Ericsson's staged approach to service management evolution enables communication service providers to take the journey at their own pace suited to their own market situation, business model and strategy. We recommend the introduction of intent-based operation only for those services and network domains that would currently profit from flexible requirement handling. Other use cases can remain as they are. In the first stage, it is possible to gain advantages from autonomous networks technology without making changes to existing service management processes. Most existing service management artifacts and processes can be reused in intent-based systems until there is a clear business case for further evolution.

The authors

Jörg Niemöller is an analytics and customer experience expert in Solution Area Cognitive Network Solutions. He joined Ericsson in 1998 and spent several years at Ericsson Research, where he gained experience in machine-reasoning technologies and their business relevance. He is currently leading the introduction of these technologies into Ericsson's portfolio to enable solutions for autonomous networks. Niemöller holds a Ph.D. in computer science from Tilburg University, the Netherlands, and a diploma degree in electrical engineering from the TU Dortmund University, Germany.

Elisabeth Müller joined Ericsson in 2006. She has taken on many roles including system design, system management and solution architecture in all business support systems (BSS) areas. Müller holds various patents within BSS and serves as a senior expert for monetization, partner and customer management, focusing most recently on service exposure architecture for the digital economy. She holds an M.Sc. in mathematics and business economics from Johannes Gutenberg University Mainz, Germany.

Massimiliano Maggiari is a senior expert in operations support systems (OSS) architecture. Since joining the company in 2006, he has held many roles across product development and product management in the OSS domain. Maggiari holds numerous patents related to OSS and control-plane-based networking, as well as an M.Sc. in electronic engineering from the University of Genoa, Italy.

Kamal Maghsoudlou has held various roles in product development and service delivery within the OSS/BSS domain in areas ranging from solution architecture to system design and management since he joined Ericsson in 2013. He has also made over 30 contributions to standardization in the areas of open API, IP services and open digital architecture (ODA), and received outstanding contributor awards from both Metro Ethernet Forum (MEF) and TM Forum. Maghsoudlou holds an M.Sc. in electronics and electrical engineering from the University of Tehran, Iran.

References

1. TM Forum, Intent in Autonomous Networks v1.3.0 (IG1253), August 15, 2022 ↗
2. TM Forum, Autonomous Networks – Reference Architecture, v1.0.1 (IG1251), July 15, 2022 ↗
3. Ericsson Technology Review, Autonomous networks with multi-layer, intent-based operation, August 31, 2023, Niemöller, J., Silvander, J., Stjernholm, P., Angelin, L., Eriksson, U. ↗
4. Ericsson Technology Review, Creating autonomous networks with intent-based closed loops, April 19, 2022, Niemöller, J., Szabó, R., Zahemszky, A., Roeland, D. ↗

Further reading

- Ericsson, Autonomous networks ↗
- Ericsson, Telecom AI ↗
- Ericsson blog, Want to get ahead in service management and orchestration? ↗
- Ericsson, E2E service orchestration ↗