ANIMA
Internet-Draft
Intended status: Informational
Expires: 3 September 2026

J. Du
Y. Tian
X. Gong
S. Jiang
BUPT

2 March 2026

# The Autonomic Deployment Mechanism of Service Intent in Autonomic Networks
draft-du-anima-service-intent-auto-deployment-00

## Abstract

This document defines a generic service intent deployment mechanism. It enables automated negotiation and coordination of heterogeneous resources. The mechanism uses RM ASAs and the Generic Autonomic Signaling Protocol (GRASP) for dynamic interactions and resource exchanges. It specifies a complete workflow covering intent reception, parsing, responder selection, negotiation, solution integration, resource confirmation, and dynamic adjustment. It employs standardized message formats, a negotiation state machine, and convergence logic to jointly optimize multiple resources and ensure end-to-end service level objectives. Its design features good scalability and fault tolerance, making it suitable for automated orchestration and lifecycle management in intent-driven networks.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 3 September 2026.

## Copyright Notice

Table of Contents



## 1. Introduction

Traditional network operation and maintenance rely on administrators manually translating business requirements into device configurations. This approach is not only inefficient but also challenging to adapt to dynamically changing network environments. As network scales expand and service types become more diverse, network administrators increasingly prefer to specify "what to achieve" rather than "how to configure devices focus from implementation details to business intent. This high-level policy expression is known as Intent, which enables the network to make autonomous decisions and dynamically adjust resources to meet business objectives.

A key challenge in intent-driven networking is translating high-level intents into concrete resource allocation actions while continuously ensuring intent satisfaction as network conditions evolve. The draft [Network-Service-Auto-Deployment] proposes a distributed resource

negotiation mechanism based on Resource Manager Autonomous Agents (RM ASAs), enabling nodes to dynamically negotiate resources such as bandwidth and queues using the GRASP protocol.This approach provides a foundation for intent-driven networking. However, the mechanism supports only value-based resource negotiation and lacks the ability to understand intent semantics. Its negotiation process focuses solely on matching resource quantities, without considering the business objectives served by the resource allocation, which hinders adaptive adjustment to changing business requirements.

This limitation is particularly evident in emerging network services that integrate transmission with computing. For example, in intelligent video compression, the network must not only reserve bandwidth resources along the forwarding path but also require specific nodes to invoke AI inference models for image compression tasks, all while ensuring end-to-end latency and image quality. Such services involve the joint optimization of network and computing resources, which are interdependent, making simple value-based resource negotiation insufficient to meet their requirements. What administrators truly need to specify are service-level objectives, such as "perform intelligent compression on video stream X while ensuring latency and image quality," rather than concrete resource allocations like "reserve 10 Mbps bandwidth on node A and 2GB memory on node B."

This document builds upon [Network-Service-Auto-Deployment] to propose a generic intent deployment mechanism based on distributed RM ASA negotiation. In this mechanism, the Service Initiator first creates a Service Intent instance as defined in draft [Service-Intent]. This instance uses a structured semantic model to specify expected network, computing, and storage resources. It may include network requirements like bandwidth, latency, jitter, endpoints, and multi-path permission; computing needs such as capacity, request latency, and coordination flags; and storage demands including capacity and throughput. The Service Initiator then uses GRASP to flood the Service Intent Objective across the autonomous domain. Upon receiving it, each Service Responder’s RM ASA starts distributed negotiations based on the intent’s semantic constraints and local real-time resource states, such as available bandwidth, GPU load, and storage I/O. Through hop-by-hop negotiation and feedback, RM ASAs collaborate to find a feasible resource allocation that meets the intent’s goals, such as end-to-end latency and image quality, ultimately enabling joint optimization of network, computing, and storage resources. This mechanism features the following characteristics:

* High-Level Abstraction: Administrators only need to express business objectives without concerning themselves with specific resource values or device configurations.

* Support for Compute-While-Transmitting: A unified framework expressing both network and computing resource requirements, enabling the dynamic deployment of computational tasks along the forwarding path.

* Distributed Autonomy: The negotiation process relies on GRASP interactions between nodes without a centralized controller, aligning with the design principles of autonomic networking.

* Intent Awareness: RM ASAs make local decisions during negotiation based on the intent, rather than relying solely on simple value matching.

* Dynamic Adaptability: When network or computing resource states change, renegotiation can be initiated to continuously ensure the satisfaction of intent.

This document defines the generic intent deployment process in Section 4. The security mechanism is based on the Autonomic Control Plane (ACP) [RFC8994] and the Bootstrapping Remote Secure Key Infrastructure (BRSKI) [RFC8995], with relevant discussions provided in Section 6.

## 2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

## 3. Terminology & Abbreviations

This document uses terminology defined in [RFC7575] and [Network-Service-Auto-Deployment].

* RM ASA: A Resource Manager Autonomous Agent operating on an autonomous node. It is responsible for managing local resources on the node, including bandwidth, queues, memory, priorities, and computing resources. The RM ASA communicates with its counterparts on other nodes to enable dynamic scheduling of network resources within an autonomous network domain. This document assumes that all autonomous nodes are equipped with an RM ASA.

* Service Initiator: An autonomous node responsible for receiving external intents and coordinating their deployment within an autonomous network. As the initiating entity of an intent, this node communicates with the RM ASAs of relevant nodes through GRASP negotiation to facilitate intent deployment and management.

* Service Responder: An autonomous node that responds to negotiation requests from the Service Initiator. It participates in the negotiation process through its RM ASA and proposes feasible resource allocation solutions based on local resource status and intent semantics.

## 4. General Deployment Mechanism Based on RM ASA Negotiation

This section describes the general process for automated deployment and negotiation of service intent based on RM ASA. This mechanism assumes that the service intent has been input by the user through an external system and converted into a standardized service intent object by a parsing module. This section defines a general negotiation framework for intent deployment and specifically specifies the message format, processing logic, and convergence mechanism for interactions between the Intent Initiator ASA and the RM ASA using the GRASP protocol. This framework is independent of specific resource types and does not presuppose the implementation method of the internal agent of the RM ASA.

### 4.1. Intent Parsing

The Service Initiator ASA receives the service intent from an external interface. The intent can be expressed in natural language or structured text, for example, "guarantee bandwidth for video conferencing between headquarters and branches and allow the use of multiple paths". The parsing module inside the Service Initiator RM ASA converts this intent into a standardized service intent object, which serves as a unified data carrier for all subsequent negotiation processes. The definition, data structure, value ranges, and extension rules of this object are detailed in [Service-Intent]; this document only references its format.

The intent object MAY contain several flags to indicate specific deployment requirements. For example, the multipath flag ‘multipath-permission‘ indicates whether multiple parallel paths are allowed; the transport-coordination flag "transport-coordination" indicates whether computing resources need to be invoked along the transmission path. During the parsing process, the Service Initiator ASA MUST perform compliance verification on the generated intent object to ensure it conforms to the draft specification and make a legality judgment based on local network policies.

### 4.2. Determination of Service Responders

To narrow the scope of subsequent negotiation, the Service Initiator RM ASA MUST determine the set of RM ASA nodes that will participate in service deployment, i.e., the Service Responder. This determination process is based on the topology information and resource overview maintained locally by the Service Initiator RM ASA. Based on the service type flag carried in the intent object, it filters forwarding nodes, computing nodes, and storage nodes in stages.

The Service Initiator RM ASA, as a node within the autonomous domain, participates in the intra-domain routing protocol to obtain and continuously maintain the network-wide topology information, thereby constructing a link state database. Simultaneously, the Service Initiator RM ASA maintains a lightweight resource status overview of each node locally, including available bandwidth, computing load, and available storage space, through the GRASP synchronization mechanism or historical data statistics. This information is stored locally on the Service Initiator RM ASA and does not rely on a centralized component.

When determining the Service Responder, the Service Initiator RM ASA first dynamically discovers potential responding nodes through the GRASP Discovery mechanism: it sends a Discovery message carrying the Resource Manager Objective, where the objective-value describes the summary requirements of the service intent (e.g., service type, required resource category). An RM ASA that receives this message and whose own capabilities meet the summary requirements MUST reply with a Response message, thus forming a preliminary candidate node set. Subsequently, the Service Initiator RM ASA parses the intent object to identify the service scenario and resource requirements. Then, combined with the locally maintained topology information and resource overview, it performs fine-grained screening of the nodes in the preliminary set to determine the final three types of Service Responders: forwarding, computing, and storage. The specific decision mechanism is as follows:

* When the ‘compute-intent‘ contains the ‘transport-coordination‘ field and its value is ‘TRUE‘, it indicates that the computing tasks in the network service need to be completed on the forwarding path. The Service Initiator ASA SHOULD consider deploying the computing tasks on nodes along the forwarding path. Specifically, the Service Initiator ASA first calculates candidate forwarding paths based on transmission constraints such as ‘destinations‘, ‘Bandwidth-Requirement‘, ‘latency-bound‘, and ‘jitter-bound‘ carried in the ‘network-intent‘ and determines the set of forwarding Service Responders that may participate on those

paths. Subsequently, the Service Initiator ASA SHOULD select, from this set, nodes that simultaneously meet the computing requirements defined in the ‘compute-intent‘ (e.g., ‘compute-capacity‘) and the storage requirements defined in the ‘storage-intent‘ (e.g., ‘storage-capacity‘, ‘storage-throughput‘), as the computing Service Responder and storage Service Responder, respectively. If no node meeting all requirements can be found, the Service Initiator MAY adjust the path selection or relax some constraints according to local policies.

* When the value of the ‘transport-coordination‘ field in the ‘compute-intent‘ is ‘FALSE‘, it indicates that the computing resources MAY exist independently of the forwarding path. In this case, the Service Initiator ASA SHOULD first select computing nodes as the computing Service Responder based on the computing requirements defined in the ‘compute-intent‘ (e.g., ‘compute-capacity‘) and possible location constraints; simultaneously, it selects storage nodes as the storage Service Responder based on the storage requirements defined in the ‘storage-intent‘ (e.g., ‘storage-capacity‘ and ‘storage-throughput‘). Subsequently, based on the locations of the selected computing and storage nodes, the Service Initiator ASA calculates the optimal forwarding paths connecting these nodes, taking into account the source and destination addresses ‘destinations‘ and transmission constraints ‘Bandwidth-Requirement‘, ‘latency-bound‘, and ‘jitter-bound‘ defined in the ‘network-intent‘. If no combination of nodes meeting all requirements can be found, the Service Initiator MAY adjust the node selection or relax some constraints according to local policies.

* If the multipath flag ‘multipath-permission‘ in the intent object is TRUE, the Service Initiator ASA SHOULD generate multiple paths that satisfy the transmission constraints and sort them by priority to support subsequent multipath negotiation; if the multipath flag is FALSE, it SHOULD select only one optimal path.

The determined results of the Service Responder are output as a candidate forwarding node list, a candidate computing node list, and a candidate storage node list for use in the step-by-step negotiation in Section 4.3. This round of process only performs preliminary screening based on resource overview, aiming to determine the scope of nodes that need to participate in the negotiation. The actual resource availability MUST be dynamically confirmed through subsequent direct negotiation with each node’s RM ASA.

If the Service Initiator ASA cannot find any Service Responder that satisfies the transmission constraints and resource requirements in the intent object, it MUST execute a fallback mechanism:

* The initiator MAY, within the scope permitted by local policy, gradually relax the value requirements for relevant fields in the ‘network-intent‘, ‘compute-intent‘, and ‘storage-intent‘ (e.g., accepting higher ‘latency-bound‘ or ‘jitter-bound‘, lower ‘Bandwidth-Requirement‘, ‘compute-capacity‘, or ‘storage-capacity‘/‘storage-throughput‘), and re-execute the screening process under the relaxed constraints. The step size, order, and termination conditions for relaxation are determined by local policy.

* If no candidate node that fully satisfies the original requirements can be generated after relaxing the constraints, the Service Initiator ASA SHOULD select the optimal approximate solution under the current resource overview according to local policies (e.g., selecting the path with bandwidth closest to ‘Bandwidth-Requirement‘ or the node with computing power closest to ‘compute-capacity‘). It SHOULD then provide feedback to the intent initiator via an out-of-band mechanism regarding the deviation between the selected solution and the original intent requirements. This feedback mechanism is outside the scope of this specification and is determined by the specific implementation.

* If none of the above mechanisms can generate any candidate nodes, the Service Initiator ASA MUST record the reason for the failure (e.g., "no available forwarding path" or "no available computing node") and be prepared to report it to the upper layer when subsequent negotiation fails.

Regardless of the fallback mechanism used, the Service Initiator ASA MUST clearly indicate in the final feedback the difference between the actual screening result and the original intent requirements, for the decision-making of the upper-layer system or administrator.

### 4.3. Negotiation with RM ASA

According to [RFC8994] and [RFC8995], all nodes in the autonomous domain have completed secure bootstrap via BRSKI and operate over the encrypted channel provided by ACP; therefore, the node identities are trusted and the communication content is secure. On this basis, any operation on resources MUST be authorized. When an RM ASA receives a negotiation request, it MUST verify the identity of the initiator and the authorization information for the requested operation based on the security context provided by ACP (e.g., checking whether the initiator is permitted to apply for a specific type of resource). If authorization verification fails, the RM ASA MUST reject the request directly and MAY carry O_DECLINE with a reason description in the M_END message. The subsequent negotiation processes in this document

all assume that the nodes have been authenticated and the operations have been authorized.

The Service Initiator ASA, as the negotiation initiator, conducts GRASP negotiation in steps with the forwarding Service Responder, computing Service Responder, and storage Service Responder based on the candidate node list determined in Section 4.2. This section defines the GRASP message type, message content, processing logic, and convergence mechanism used in the negotiation process. The negotiation interaction follows the [RFC8990] specification, and all messages are serialized using CBOR.

#### 4.3.1. GRASP Message and Objective Definition

The negotiation process uses the following GRASP message types:

* M_REQ_NEG (3): Initiate a negotiation request.

* M_NEGOTIATE (5): Conduct multiple rounds of negotiation and exchange proposals.

* M_WAIT (7): Request the peer to wait, extending the timeout.

* M_END (6): End the negotiation, carrying an option of acceptance (O_ACCEPT) or decline (O_DECLINE).

All messages MUST carry the ‘session-id‘ field, which is generated by the Service Initiator ASA at the start of each negotiation session and is used to associate all messages of the same session. The ‘session-id‘ MUST be a random number, and its collision probability SHOULD be extremely low.

This document uses the ‘Resource Manager‘ Objective defined by [Network-Service-Auto-Deployment], and only need to supplement the intent information and resource reservation status information carried in its ‘objective-value‘. Its format is as follows:

```
objective-value = autonomic-network-service-value

autonomic-network-service-value = [
    [ service-type, service-id, service-lifetime, service-tag ],
    [* resource-requirement-pair ],
    [ intent-des, proposal-list ]
]
```

Figure 1: Extended Format of Resource Manager’s objective-value

In the request message, the ‘objective-value‘ include ‘intent-des‘, which is the original service intent (including network, computing, and storage requirements). The intent-des object contains three optional members shown as follow. The ‘network-intent‘, ‘compute-intent‘, and ‘storage-intent‘ objects are defined in [Service-Intent] From the Service Responder list obtained in Section 4.2, each node falls into one of three categories: storage, computing, and forwarding responders. The intent initiator MAY construct different ‘intent-des‘ based on the responder type. For example, for a computing Service Responder, the ‘intent-des‘ contains the ‘compute-intent‘ field information from the original intent.

```
intent-des = {
    ? "network" : network-intent,
    ? "compute" : compute-intent,
    ? "storage" : storage-intent
}
```

Figure 2: Service Intent Content Structure

In the response message, the ‘objective-value‘ include ‘proposal-list‘, which is the list of feasible solutions generated by the RM ASA. The format of ‘proposal-list‘ is defined as follows:

```
proposal-list = [* proposal]
proposal = {
  "proposal-id": uint,           ; Unique identifier of the proposal
  "guaranteed-resources": {      ; Amount of resources guaranteed
    ? "bandwidth": uint,          ; bps
    ? "latency": uint,            ; ms (contribution to latency)
    ? "compute-capacity": uint,   ; GFLOPS
    ? "storage-capacity": uint,   ; GB
    ? "storage-throughput": uint, ; MB/s
    ...
  },
  "expected-contributions": {     ; Expected contributions to end-to-end SLOs
    ? "added-latency": uint,       ; ms
    ? "added-jitter": uint,        ; ms
    ...
  },
  ? "cost": uint,                  ; Optional cost metric
  ? "expiry-time": uint            ; Validity period of the proposal (milliseconds)
}
```

Figure 3: Format of proposal-list

If an RM ASA cannot provide any feasible solution, it SHOULD carry the O_DECLINE option in the M_END message and MAY attach a reason string.

#### 4.3.2. Negotiation Process

The negotiation process is divided into the following steps. The Service Initiator ASA needs to initiate negotiation sessions for each candidate node sequentially or in parallel. For clarity, the following description takes a single Service Responder as an example.

Step 1: Initiate a Negotiation Request. Service Initiator ASA sends an M_REQ_NEG message to the target RM ASA. The message format is:[M_REQ_NEG, session-id, objective] Where:

* session-id: Newly generated random number.

* objective: Contains the GRASP Objective named ‘Resource Manager‘, ‘loop-count‘ initially set to GRASP_DEF_LOOPCT, and ‘objective-value‘ as the original service intent object ‘intent-des‘.

After sending the message, the Service Initiator ASA starts a negotiation timer with an initial value of GRASP_DEF_TIMEOUT (default 60000 milliseconds). If the timer expires before a response is received, it is considered a negotiation failure, and the initiator MAY attempt retransmission (using exponential backoff) or abandon the node.

Step 2: RM ASA Processes the Request and Responds. After receiving the M_REQ_NEG message, the RM ASA performs the following operations:

* Parse the ‘objective-value‘ to obtain the resource requirements (bandwidth, latency, computing capacity, etc.) from the service intent. The ‘objective-value‘ contains ‘intent-content‘, which MAY involve multiple types of intent expressions such as ‘storage-intent‘, ‘compute-intent‘, and ‘network-intent‘. The resource content to be negotiated varies depending on the type.

* Based on the negotiation intent type, it queries the corresponding local real-time resource status for that intent and generates one or more feasible resource allocation proposals. Each proposal MUST include the amount of resources guaranteed and the expected contribution to end-to-end performance (e.g., added latency).

* If at least one proposal is generated, it constructs an M_NEGOTIATE message to return; if it cannot generate any proposal that meets the tolerance, it constructs an M_END message (carrying the O_DECLINE option) to return.

Upon a successful response, the Service Responder RM ASA sends an M_NEGOTIATE message in the format [M_NEGOTIATE, session-id, objective], where the objective-value is a proposal-list containing all feasible proposals. Upon failure, it sends an M_END message in the format [M_END, session-id, [O_DECLINE, ?reason]], where reason is an optional UTF-8 string explaining the cause. After sending M_NEGOTIATE, the RM ASA SHOULD temporarily reserve the resources involved in the proposed proposals. However, the reservation time SHOULD NOT be too long (it is RECOMMENDED that it does not exceed GRASP_DEF_TIMEOUT) to prevent deadlock.

Step 3: Service Initiator ASA Processes the Response. After receiving the response, the Service Initiator ASA, if it receives an M_END with O_DECLINE, records the node's rejection and attempts the next candidate node. If it receives an M_NEGOTIATE, it parses the 'proposal-list', extracting the resource guarantees and contribution values of each proposal. The Service Initiator ASA MUST evaluate whether these proposals are acceptable based on the end-to-end constraints.If the currently received list of proposals is insufficient to meet the end-to-end constraints, the Service Initiator ASA MAY initiate multiple rounds of negotiation, sending a new M_NEGOTIATE message, adjusting the request, and decrementing the 'loop-count'. This process MAY be iterated multiple times until an agreement is reached or the negotiation is terminated.

Step 4: Multiple Rounds of Negotiation and Convergence. If the Service Initiator ASA wishes to adjust the request, it MAY send an M_NEGOTIATE message, where the 'objective-value' MAY contain an updated 'intent-des' or specify the proposal ID it wishes to negotiate. Upon receiving the M_NEGOTIATE message, the RM ASA SHOULD regenerate the proposal list and ensure that the 'loop-count' is decremented. If the 'loop-count' is reduced to 0, it MUST stop the negotiation and return an M_END with O_DECLINE. If either party needs more time to process during the negotiation, it MAY send an M_WAIT message in the format [M_WAIT, session-id, waiting-time], where 'waiting-time' is the suggested extension time (in milliseconds) for the peer to wait. Upon receiving an M_WAIT, the peer SHOULD reset the negotiation timer to 'waiting-time' and continue waiting.

Step 5: Negotiation Termination. When both parties reach an agreement, the Service Initiator ASA sends an M_END message carrying the O_ACCEPT option, in the format [M_END, session-id, [O_ACCEPT]]. Upon receiving the M_END message, the peer MUST immediately terminate the negotiation session and release temporary resources (unless the resources have been confirmed and reserved). If the negotiation fails due to timeout or exhaustion of the 'loop-count', both parties MUST release all temporarily reserved resources.

Step 6: Coordination of Multi-Node Negotiation. The Service Initiator ASA needs to conduct the above negotiations with the forwarding, computing, and storage nodes sequentially or in parallel. The following order is RECOMMENDED:

* When the ‘compute-intent‘ contains the ‘transport-coordination‘ field and its value is ‘TRUE‘, it is RECOMMENDED to first negotiate the path with the forwarding node, then negotiate computing resources with the computing nodes on that path, and finally negotiate storage with the storage node.

* When the ‘compute-intent‘ contains the ‘transport-coordination‘ field and its value is ‘FALSE‘, it is RECOMMENDED to first negotiate with the computing node and storage node, and then negotiate the path with the forwarding node based on the locations of the selected nodes.

For each candidate path, the Service Initiator ASA MUST collect responses from all relevant nodes on that path and proceed to the proposal integration described in Section 4.4. If the current path negotiation fails, it SHOULD release all temporarily reserved resources on that path and attempt the next candidate path.

## 4.4. End-to-End Coordination and Solution Integration

After collecting the responses from all Service Responders on the current path, the Service Initiator ASA MUST perform end-to-end coordination. The specific steps are as follows:

* Select one proposal from the proposal list of each responding node to form a complete end-to-end resource configuration chain. Ensure that the selected proposals are compatible with each other in terms of resource types (e.g., the bandwidth committed by the forwarding node matches the transmission bandwidth required by the computing node).

* If multiple combinations of proposals meet the requirements, the Service Initiator ASA SHOULD select the optimal combination based on local policies (e.g., minimum cost, lowest latency). If the current path supports multipath, the Service Initiator ASA SHOULD distribute traffic across multiple paths and MUST ensure that the combined performance of each path meets the overall service level objective.

* Once a combination of proposals that meets the conditions is determined, the Service Initiator ASA MUST send a GRASP M_END message (carrying O_ACCEPT) to all relevant RM ASAs on the path to formally confirm the selected proposals and reserve the resources.

Upon receiving the confirmation, each RM ASA SHOULD convert the temporary reservation into a formal reservation and persistently store the binding information.

* Regardless of whether the deployment is successful or not, the Service Initiator ASA MUST return a standardized feedback message to the Service Initiator. This feedback MUST include the final deployment result; if successful, it MUST detail the actually allocated resource details (including the actual path, the amount of resources committed by each node, and the expected end-to-end performance); if failed, it SHOULD provide the reason for the failure.

If the current path negotiation fails, the Service Initiator ASA MUST release all temporarily reserved resources and attempt the next candidate path. If all candidate paths fail to meet the requirements, the Service Initiator ASA MUST report the intent deployment failure to the upper layer.

### 4.5. Dynamic Maintenance and Adjustment

During the intent lifecycle, if a Service Responder RM ASA cannot maintain the original proposal due to changes in local resources, it MAY actively initiate renegotiation. The renegotiation process is as follows:

* The RM ASA sends an M_REQ_NEG message to the Service Initiator ASA, carrying the original intent object "intent-des", indicating the need for renegotiation.

* Upon receiving this message, the Service Initiator ASA triggers the renegotiation process. It MUST re-collect the latest feasible proposals from each node on the current path following the steps in Section 4.3 and re-integrate them.

* If the original proposal cannot be maintained, an attempt MAY be made to adjust the allocation or switch to another candidate path.

In addition, the Service Initiator ASA MAY periodically evaluate link quality and proactively initiate renegotiation to seek a more optimal resource allocation scheme, thereby ensuring the continuous fulfillment of service level objectives. During renegotiation, the original resource allocation remains effective until the new negotiation is successfully confirmed. If negotiation for increased requirements fails, the original allocation remains unchanged, and the responder MUST NOT revoke it; if negotiation for reduced requirements succeeds, the excess resources are released; if the negotiation is terminated, both parties revert to the original allocation.

### 4.6. Intent Termination and Resource Release

When the service intent lifecycle ends or is externally revoked, the Service Initiator ASA MUST actively release the occupied resources. To achieve this, the Service Initiator ASA sends an M_REQ_NEG message to all relevant Service Responder RM ASAs, in which the carried intent object MUST have all resource requirements set to zero. Upon receiving this message, each RM ASA MUST confirm the release of local resources and reply with an M_END (O_ACCEPT).

If the intent resource has a lease period, each RM ASA MUST automatically release the local resources when the lease expires and MAY send an M_REQ_NEG carrying a resource status change notification to the Service Initiator ASA, informing that the resources have been released or the status has changed. Upon receiving the notification, the Service Initiator ASA updates the locally maintained intent status and MAY provide feedback to the upper-layer system that the resources have expired.

To prevent resource leaks caused by the failure of the Service Initiator ASA, each RM ASA MUST have a timeout mechanism for temporarily reserved resources and automatically reclaim the resources after the timeout. Formally reserved resources SHOULD also have a lease period. When the lease is half expired, the Service Initiator ASA MAY initiate a lease renewal negotiation. If the lease is not renewed in time, the RM ASA MUST automatically release the resources upon lease expiration and notify the initiator.

## 5. IANA Considerations

This document reuses the ‘Resource Manager‘ GRASP objective name defined in [Network-Service-Auto-Deployment] and does not request a new objective name. This document extends the objective value of this objective to carry the intent-object; however, the specific format of the intent-object is defined by an external specification and is out of scope of this document. This document does not request

the establishment of new IANA registries.

## 6. Security Considerations

The mechanisms described in this document fully rely on the secure communication environment provided by the Autonomic Control Plane (ACP) [RFC8994] and comply with the security specifications of the Generic Autonomic Signaling Protocol (GRASP) [RFC8990]. All autonomous nodes participating in intent negotiation must complete the Bootstrapping Remote Secure Key Infrastructure (BRSKI) [RFC8995] before joining the autonomous domain, to obtain a trusted identity and establish a secure control-plane channel.

The intent injection and parsing process involves an external management interface, which should implement strict identity authentication and authorization mechanisms to prevent unauthorized intent issuance. After receiving an external intent, the RM ASA may perform validity verification according to local policies to ensure the intent is within the allowed scope.

During GRASP negotiation, all messages carrying the intent-object are transmitted encrypted via ACP, ensuring confidentiality and integrity of intent content and negotiation results. Inter-node negotiation interactions shall follow GRASP’s anti-replay and anti-tampering mechanisms to prevent malicious nodes from forging or modifying negotiation messages.

When intent involves resource reservation and release, the RM ASA must ensure the authorization of resource operations, responding only to requesters authenticated via ACP with corresponding permissions. After the intent lifetime ends, resources must be released promptly to prevent resource leakage and malicious exploitation.

Overall, the security and reliability of the mechanisms described in this document are built upon the existing security infrastructure of ACP, BRSKI, and GRASP. Any deployment must ensure that these foundational security measures are fully implemented.

## 7. References

### 7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <https://www.rfc-editor.org/info/rfc7575>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/info/rfc8990>.

[RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <https://www.rfc-editor.org/info/rfc8994>.

[RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <https://www.rfc-editor.org/info/rfc8995>.

## 7.2. Informative References

[Network-Service-Auto-Deployment]
Jiang, S., Ed. and Z. Du, "A Generic Autonomic Deployment and Management Mechanism for Resource-based Network Services", Work in Progress, Internet-Draft, draft-ietf-anima-network-service-auto-deployment-07, 2026, <https://datatracker.ietf.org/doc/draft-ietf-anima-network-service-auto-deployment/>.

[Service-Intent]
Wang, B., Zhu, L., and S. Jiang, "Definition of Service Intent in Autonomic Networks", Work in Progress, Internet-Draft, draft-zhu-anima-service-intent-00, 2026, <https://datatracker.ietf.org/doc/draft-zhu-anima-service-intent/>.

## Authors' Addresses

Jialu Du
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Beijing
Haidian District, 100083
China
Email: dujialu2024@bupt.edu.cn

Ye Tian
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Beijing
Haidian District, 100083
China
Email: yetian@bupt.edu.cn

Xiangyang Gong
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Beijing
Haidian District, 100083
China
Email: xygong@bupt.edu.cn

Sheng Jiang
Beijing University of Posts and Telecommunications
No. 10 Xitucheng Road
Beijing
Haidian District, 100083
China
Email: shengjiang@bupt.edu.cn