This PDF is generated from authoritative online content, and is provided for convenience only. This PDF cannot be used for legal purposes. For authoritative understanding of what is and is not supported, always use the online content. To copy code samples, always use the online content.

SIP Cluster Solution Guide

Agent Reservation in SIP Cluster

5/3/2025

Contents

Agent Reservation in SIP Cluster

Agent Reservation is a very important concept in SIP Cluster. The Agent Reservation flow is illustrated with the following sample diagram:

In the above diagram, Interaction Server handles multimedia interactions for agents. Interaction Server is connected to all routing Stat Servers and reports agent states to all routing Stat Servers in the environment. All routing Stat Servers are aware of the activities of the multimedia agents. Interaction Server is connected to all URS instances in the SIP Cluster and this allows Interaction Server to load balance the routing requests across all available URS instances.

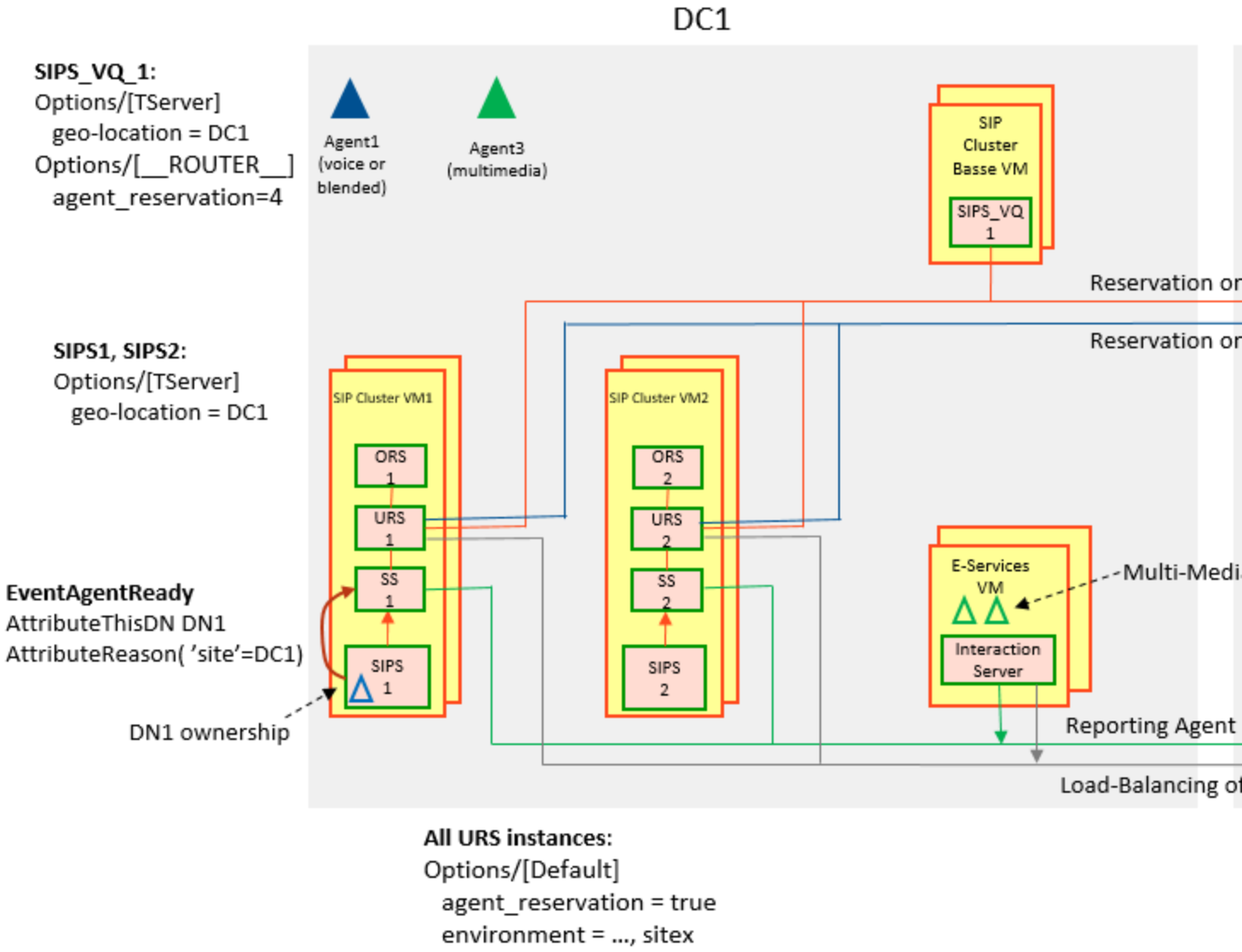
Voice calls arriving to the SIP Cluster node are processed by routing components installed on this node. Agents handling voice calls are logged into the SIP Cluster and their locations are reported to all routing Stat Servers.

A pair of VQ SIP Server instances runs in regular non-cluster mode and is used for agent reservation. URS selects a VQ SIP Server for sending a TRegisterAgent request based on the following configuration:

- If routing is requested to local agents in the routing application (the **environment** option is set in URS), URS selects the VQ SIP Server collocated with the agent, based on the VQ SIP Server's **geo-location** setting.
- If routing is requested without specific agent location or the agent location is unavailable, URS selects the VQ SIP Server based on its priority that is configured in the **agent_reservation** option, the **[__ROUTER__]** section, of the VQ SIP Server.

Reservation Based on Agent Location

The following diagram illustrates reservation based on the agent location:



Agents handling voice calls have DN ownership assigned in the SIP Cluster node. DN ownership defines the agent location. SIP Server reports the agent location in EventAgentReady in the **site** key-value pair of AttributeReason. The value of the **site** key is taken from the application-level **geo-location** option of the VQ SIP Server.

When the **environment** option is explicitly set to `sitex` in URS, URS selects the VQ SIP Server for sending the TReserveAgent request based on VQ SIP Server's **geo-location** setting. In the diagram:

- All requests for Agent 1 reservation go to SIPS_VQ_1, which has **geo-location** set to `DC1`.
- All requests for Agent 2 reservation go to SIPS_VQ_2, which has **geo-location** set to `DC2`.

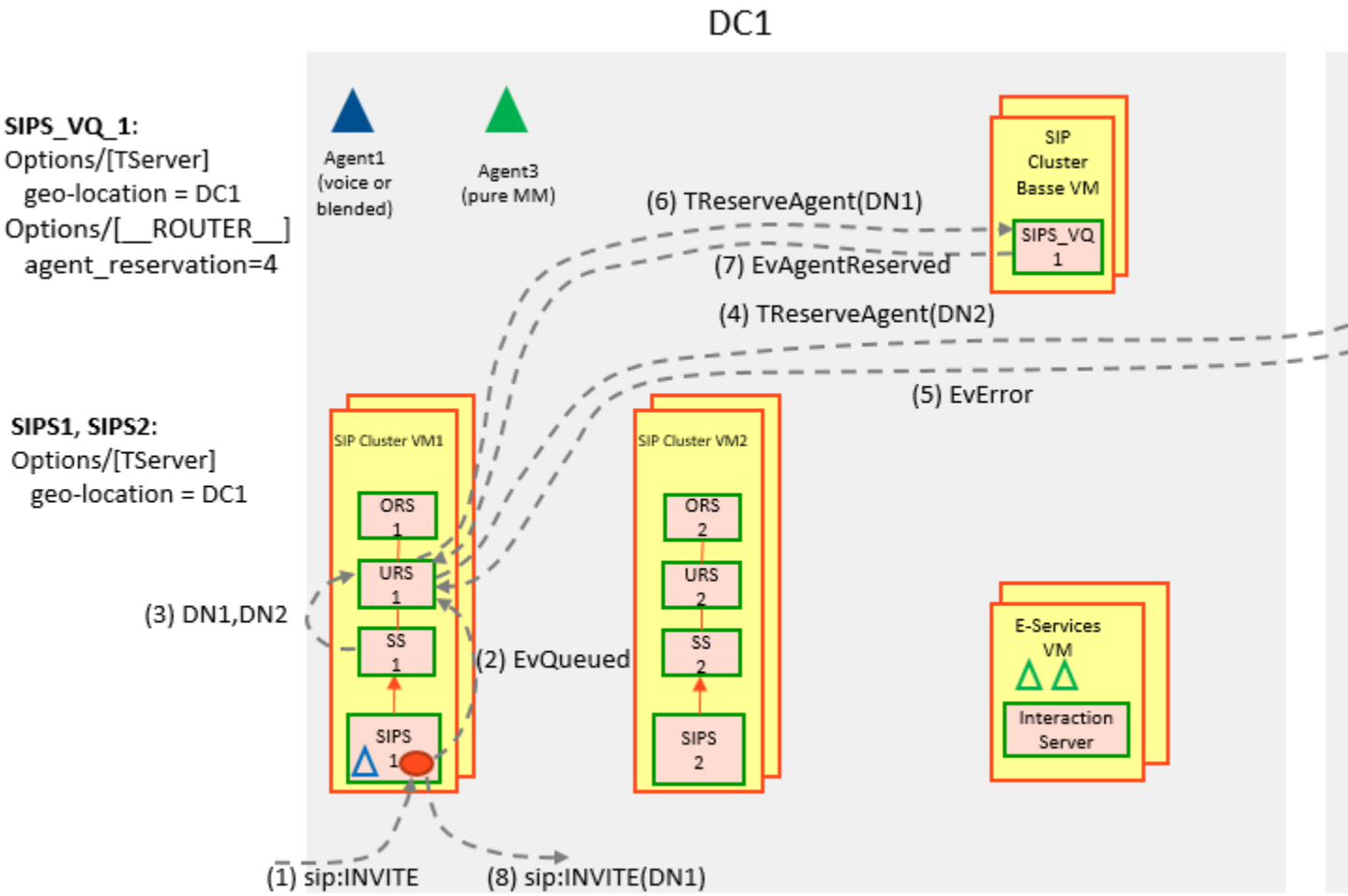
Agents handling multimedia interactions don't have DN ownership assigned. Interaction Server doesn't report the agent location. URS selects the VQ SIP Server for sending the TReserveAgent request based on priority that is specified in the **agent_reservation** option, the **[__ROUTER__]**

section of the VQ SIP Server. In the diagram, the reservation priority of SIPS_VQ_1 is 4, the higher digit the higher priority. So all reservation requests are sent to SIPS_VQ_1. If SIPS_VQ_1 is unavailable, reservation requests are sent to SIPS_VQ_2.

Agent Reservation Conflicts

Agent Reservation conflicts occur when there are multiple URS instances trying to route a call to the same agent.

The following diagram illustrates an example of the reservation conflict:



In the above diagram, there is a call on SIP Server 1 and another on SIP Server 3. The calls are sent to URS 1 and URS 3. The routing Stat Servers on both nodes provide the same set of available DNs (DN1 and DN2). Both URS instances select SIPS_VQ_2 based on the location of Agent 2. The request

from Node 3 is granted and the request from Node 1 is rejected. Node 1 makes another reservation attempt to DN1, which is granted.

Agent reservation conflicts are normal in a SIP Cluster environment where multiple nodes route different calls using the same strategy and target the same group of agents. In the SIP Cluster, conflicts cannot be avoided completely due to the distributive nature of the system, but they can be minimized as discussed in the following section.

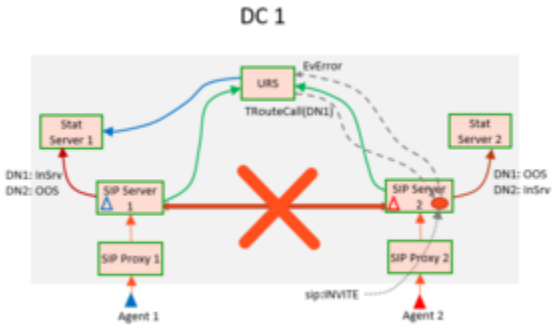
Centralized Routing for Reducing Conflicts

Switching to centralized routing reduces agent reservation conflicts. Consider the following sample architecture diagram:

In the above diagram, there is a centralized URS HA pair that serves the entire data center. There is no separate URS instance serving each node. The fewer instances of URS you use, the lesser the number of reservation conflicts.

URS is connected to the default port that processes local calls from SIP Server 1 and SIP Server 2. URS is also connected to the Stat Server and the Stat Server is connected through the T-Controller layer, which provides URS a global view of all agents and their availability in the SIP Cluster.

However, in a centralized routing scenario, problems might occur when the connection between SIP Cluster nodes is lost and URS no longer has the complete DN availability view. As shown on the following sample diagram, URS is connected to Stat Server 1 and selects DN2 to route a call on Node 2. As a result, URS routes the call to a DN that is out of service, which leads to a failed routing attempt.



Node-Based Routing vs. Centralized Routing

The following table summarizes node-based vs. centralized routing options, their benefits and limitations.

	Node-Based Routing	Centralized Routing
Pros	• Better isolation and modularity (URS maintenance on one node does not affect other nodes) • Better reliability • Better suited for automated deployment	• Fewer agent reservation conflicts (number of conflicts increases with an addition of new data centers or new URS instances in a data center) • Fewer components to deploy
Cons	• Number of agent reservation conflicts grows with the number of nodes	• Less reliable than node-based routing • In case of multiple URS instances per data center, manual pairing of URS and SIP Cluster nodes is required.

Location-Based Routing

Location-based routing is another way to effectively minimize agent reservation conflicts. As discussed in the previous sections, URS has the ability to select a VQ SIP Server based on the agent location using the **site** key-value pair of AttributeReason in EventAgentReady.

Consider a sample scenario where the EventReadyAgent and EventQueued/EventRouteRequest events have the following values:

Location of a ready agent	**Location of a call**
EventAgentReady *AttributeReason* '**site**' '**uswest**' *AttributeThisDN* '7770002' **AgentSessionID** '01M38P5D08AA...' *AttributeAgentWorkMode* 0 *AttributeAgentID* '7770002'	**EventQueued/EventRouteRequest** *AttributeCallType* 2 *AttributeConnID* 0151025a84c2a028 *AttributeDNIS* '5001' *AttributeANI* '8880001' *AttributeThisDN* '5001' *AttributeThisDNRole* 2 *AttributeExtensions* 'OtherTrunkName' 'Trunk_SBC_PSTN_us-west-1' '**geo-location**' '**uswest**'

In the above sample, the value of **site** is populated as *uswest*. When a call arrives at the SIP Cluster node, in the EventQueued/EventRouteRequest messages in the AttributeExtensions, there is a **geo-location** key-value pair. The value of **geo-location** is the same as the **site** key. You can enforce URS to process matching these parameters by creating a skill expression using the **sitex()** function. For example, the routing strategy in Composer can use the following skill expression to route the call to only the agents logged-in to the local data center:

```
sitex(GEO_LOCATION) & English > 0
```

This approach helps to reduce the number of agent reservation conflicts. **geo_location** is passed as a parameter to the **sitex** function. In this case, the reservation requests for agents handling voice calls are load-balanced across data centers and the VQ SIP Server's reservation requests load is reduced.

Composer supports the **sitex()** function. Workflows in Composer provide access to extension data (xdata) to read the geo-location. The Target block accepts a skill expression that can include the **sitex()** function.

Next topic: Routing Optimization