Universal Routing Reference

Using Agent Skills for Ideal Agent Selection

5/6/2025

Contents

- 1 Using Agent Skills for Ideal Agent Selection
  - 1.1 SetIdealAgent
  - 1.2 Example Skill Expression
  - 1.3 How the Skill Expression is Used
  - 1.4 Calculated Deviation
  - 1.5 Calculating the Deviation From Ideal
  - 1.6 Multiple Interactions Competing for Same Agent
  - 1.7 Weighted Agent Skills
  - 1.8 If Strategy Does Not Use SetIdealAgent
  - 1.9 Important Notes About Ideal Agent

Using Agent Skills for Ideal Agent Selection

Starting with URS 8.1.400.19, Universal Routing can select the most ideal agent to handle an interaction when more than one agent is available. You can also use this functionality to select the most ideal interaction when there is more than one interaction competing for the same agent. To implement this functionality, this release introduces two new functions, `SetIdealAgent`, `TargetListSelected` and new option `set_ideal_agent`.

SetIdealAgent

Parameter: Skill Expression: `STRING` (constant or variable representing a skill expression)

Return value type: `VOID`

This function is available from IRD's **Function** object, **Target Manipulation** category. The `SetIdealAgent` function utilizes a "best fit" factor as one criteria for selecting the most ideal agent to handle an interaction or to select the most ideal interaction when there is more than one interaction competing for the same agent. Once you define the ideal agent's skill set, Universal Routing Server will use this definition if there is any choice when assigning agents to interactions.

Example Skill Expression

The `SetIdealAgent` function accepts as input a skill expression that you define via the **Skill Expression** field in the Function object dialog box. Clicking opposite **Skill Expression** opens the **Skill Expression Properties** dialog box where you create the skill expression. Genesys recommends that functions not be used in ideal agent skill expressions and that the skill expression be in DNF form, such as:

```
Skill1=value1 & Skill2>=value2 & Skill2<=value3 | Skill3=value3
```

URS would interpret the above expression as follows: The ideal agents are those who have Skill1 set to value1 (`Skill Level` in Configuration Database) and Skill2 in between value2 and value3 or alternatively agents with Skill3 set to a value of 3.

Other examples:

```
Skill_A < 3 | Skill_A < 6 & Skill_C > 3 & Skill_D >= 6
Skill_A > 3 & Skill_B > 3 & Skill_C > 3 & Skill_D < 3
Skill_A >= 2 & Skill_B >= 3 & Skill_C >= 4
```

For information on creating skill expressions, see the *Universal Routing 8.1 Interaction Routing Designer Help*. Go to Creating a New Strategy > Expression Building. For more detailed information, search on "Skill Expression" in the *Universal Routing 8.1 Reference Manual*.

How the Skill Expression is Used

When processing interactions using the ideal agent definition, URS measures every qualified agent for the interaction being processed to determine how exactly the agent's skills match the ideal agent skills. An extra metric is associated with every agent (see `TargetListSelected` below), which indicates how close the agent is to the ideal agent definition. URS then selects the agent whose skills deviate the least from the skills of the ideal agent.

For more information on this deviation, see Multiple interactions Competing for Same Agent.

Calculated Deviation

A new function, `TargetListSelected`, returns the calculated deviation of the selected agent from the ideal agent.

TargetListSelected

Parameter: Key: `None`

Return value type: `STRING`

This function returns information about the selected target. If used after the function `SetIdealAgent`, it will return additional key `mismatch` with the deviation value of the selected agent. This function also returns much of the same data that functions SelectDN/SuspendForDN return about the selected target as described in the *Universal Routing 8.1 Reference Manual*.

Calculating the Deviation From Ideal

URS calculates the deviation of an agent from the ideal agent in following way:

If the comparison operation evaluates to a true score of 0, the agent skill does not deviate from the ideal agent skill. A score greater than 0 indicates how much the agent skill is different from the ideal agent skill.

- Use of the & operator in the skill expression adds to the score.
- Use of the | operator in the skill expression results in a minimal score.

URS pre-calculates agent skill deviations at the moment some queue is created, not at the moment when an agent or interaction is actually selected. That means that skill expressions used to define the ideal agent must contain only skills, numbers, and logical/comparing/mathematical operations. URS tracks skill updates in the Configuration Database and will recalculate deviations if a skill update occurs. Genesys recommends not using functions in skill expressions. URS does not track when a function value changes, which could result in the wrong deviations being used.

Example Deviation Calculation

Agent Skills:

AGENTS	SKILL A	SKILL B	SKILL C	SKILL D
Agent 1	1	2	3	4
Agent 2	2	3	4	
Agent 3	3	4	5	6
Agent 4	4	5	7	1
Agent 5	5	6	1	2
Agent 6	5	6	1	2

Deviation for the Skill_expression:

```
Skill_A >=2 & Skill_B <=5 & Skill_C=5
```

AGENTS	SKILL_A >=2	SKILL_B <=5	SKILL_C = 5	SKILL_D	DEVIATION
Agent 1	1	0	2	0	3
Agent 2	0	0	1	0	1
Agent 3	0	0	0	0	0
Agent 4	0	0	2	0	2
Agent 5	0	1	4	0	5
Agent 6	0	1	4	0	5

Agents with Same Deviation

If an ideal agent is set for an interaction and there are multiple available agents, then URS checks every one of them and selects the agent having the smallest deviation. If deviations are the same, then URS uses the value of a statistic to select an agent.

Ideal agent selection is agent-level selection and works as described above only when URS uses agent-level statistics. In this case, strategy targets are based on skill expressions or are the result of function `UseAgentStatistics[true]` being called in the strategy.

Agent/Place Groups

If a strategy uses Group-level statistics (URS uses statistics to select the best Agent Group or Place Group), then URS behaves as follows:

- URS uses the ideal agent skill expression to select an agent inside every participating Agent Group.
- If more than one Agent/Place Groups has available agents, then URS uses Group statistics to decide which group to use. URS selects based on the Group having the best statistics, not the group having the most ideal agents.

Specifically, this means if routing targets are sets of Agent Groups, URS uses `UseAgentStatistics[true]` to select the most ideal agent throughout all listed Agent Groups. Note that because the default value of this function is `false`, when creating the strategy, you must set `UseAgentStatistics` to `true` and position the function before the Agent Group target.

Multiple Interactions Competing for Same Agent

Note: Apart from step 2, all the steps listed below are the standard steps that URS always executes.

URS can also use skill expressions defined for function `SetIdealAgent` when there are multiple interactions in queue competing for the same agent (desired agent). As an example, assume an agent becomes available and there are two competing interactions for this agent (Call1 from VQ 1 and Call2 from VQ2). In this case:

1. URS checks interaction priority. You can use various functions (`Priority`, `IncrementPriority`, `SelectDN`, `SetVQPriority`, and so on) in a strategy to specify interaction priority for different queues. If none of the priority functions is used, then interaction priority is 0. If one of the priority functions is used, the interaction with the higher priority will be used. If priorities are the same, go to step 2.
2. URS checks "best fit" factor. URS only executes this step if both interactions have ideal agent set.

- *URS will select the interaction with the ideal agent skill set that is closest to the skill set of the desired agent.*
- **Note:** There may be some cases where at least one of the interactions will not have ideal agent set, or cases where the ideal agent definition for both interactions is equally close to the desired agent. For these cases, URS checks timing as described in step 3.

- When URS checks timing, the following values are calculated: `X1 = T1 + D1 and X2= T2+D2.`

The example below uses voice interactions.

- T1 - The time Call1 has been waiting for this agent (with milisecond precision)
- T2 - The time Call2 has been waiting this agent (with milisecond precision)
- D1 - If the strategy for Call1 instructs to use prediction ("what-if" wait time where function `PriorityTuning` is used with Prediction set to true), then it is prediction time; otherwise it is 0.
- D2 – If the strategy for Call2 instructs to use prediction ("what-if" wait time where function `PriorityTuning` is used with `Prediction` set to `true`) then it is prediction time; otherwise it is 0. Go to step 3b.

3b.If both interactions have a risk of not meeting the Service Objective, then

`X1= X1/Call1ServiceObjective` and `X2= X2/Call2ServiceObjective`.

If at least one of the interactions does not use Service Objective, this step is skipped (values X1 and X2 are not changed). Go to 3c.

3c.If X1 is bigger than X2, Call1 is used and vice versa. If (in the rare case) X1 and X2 are still the same, go to step 4.

4. URS keeps a global counter. Every time an interaction is placed into a waiting queue, the counter is incremented by one (no two different entries of some interaction into some queue can have the same

value). URS will select the interaction having the smaller counter.

Weighted Agent Skills

Starting with URS 8.1.400.22, you can apply a weight to any skill (or combination of skills) to increase the importance of a skill. Expressions can include both skill functions and arithmetic operations:

+ (plus)
- (minus)
/ (divide)
and
* (multiply)

URS considers the value of this expression as a deviation from the ideal agent (the less its value, the more ideal the agent).

There are no restrictions to using these operators, other than that the final skill expression must be a well-formed mathematical expression having the right balance of opening and closing brackets.

Weighted Ideal Agent Samples

For example, the following skill expressions show how mathematical expressions can increase or decrease a deviation and define the importance of a specific skill:

```
5*(Skill1=value1) & Skill2>=value2 & 4+(Skill2<=value3)
(Skill1=value1)-2 & 3*(Skill2>=value2 & Skill2<=value3)
3*(Skill2>=value2) & (Skill2<=value3)/2
```

The skill expression below is an example where an agent with an `employeeid` starting with `ABC` is 10 times better other agents (the deviation will be 0 for such an agent and 10 for all others).

```
10*(name(“ABC*”)=1)
```

The `name` and other skill expression functions are available in IRD's Skill Expression Properties dialog box.

If Strategy Does Not Use SetIdealAgent

Option `automatic_ideal_agent` can be used as an alternative to using `SetIdealAgent`.

automatic_ideal_agent

Location in Configuration Layer by precedence: Routing Point, T-Server, Tenant, URS

Default value: `false`

Valid values: `true`, `false`

Value changes: take effect immediately

If set to `true`, then when URS places the interaction into a queue for first time:

- If this queue targets/agents are defined as a skill expression and
- if function SetIdealAgent was not yet called for this call, then

URS will automatically call the `SetIdealAgent` function with the value of the skill expression used for this queue as a target.

Important Notes About Ideal Agent

- When URS evaluates routing targets, the use of the `SetIdealAgent` function adds an extra layer to the evaluation process. Except for the interaction priority layer, the Ideal Skill layer will take precedence over other layers defined in a routing strategy. As a result, use of the `SetIdealAgent` function in a strategy can possibly affect (for good or bad) existing routing solutions, such as Service Objective routing, Age of Interaction routing, and so on. If your routing strategy combines different target selection criteria, Genesys recommends that you analyze how these different types of target selection criteria will interact and coexist with each other.
- The first time URS executes a strategy, statistics are normally not yet opened. As a result, if the target selection is statistic or skill-based, the first interaction could be distributed to any ready agent.
- Selection of an ideal agent for an interaction is effectively agent-level selection. It works as described here only when URS uses agent-level statistics: either when targets are skill expressions or when function `UseAgentStatistics[true]` is called in strategy.
- The interaction selection criteria associated with the SetIdealAgent function are only supported in a multi-URS environments where the same target might be selected by different instances of URSs if:
  all URS instances have the same value of option `automatic_ideal_agent`
  or
  all strategies running/served by URSes include function `SetIdealAgent` (with not empty parameter).

Important

Applying different routing priority criteria (Service Level, Ideal Agent) to calls waiting for the same targets might sometimes result in out of order routing. In this case, URS records a *composite priority misused* message in the log.