Voice Callback and URS

VCB Implementation

12/14/2025

Contents

VCB Implementation

Standard URS Behavior

In a call surplus Scenario,

1. Agent becomes ready.
2. An *agent becomes ready* event is received from Stat Server.
3. URS builds a queue of all calls waiting for this agent based on priority, waiting time, etc.
4. URS goes through this queue in search of the first *routable* call (starting from the very first call until either it finds such a call or the end of the queue is reached).
5. Once (if) such a call is found, it is routed to that agent.
6. Processing of the *agent becomes ready* event is complete.

Important

A call is routable if nothing prevents the call from being routed to the agent right now – all thresholds satisfied, all criteria fulfilled, and call is in proper state (that is, selectable, no route delay, etc.).

Step 1: Adding VCB, First Implementation (8.1.400.07)

The first implementation is built on top of the standard URS behavior. Extra functionality is quite simple. While going through the queue of calls waiting for the ready agent and looking for first routable one, URS also looks for the first **VCB notifiable** call. If found, such a call is remembered. By definition, it will have a higher priority than all other **VCB notifiable** calls as well as routable calls, because the call is in the queue ahead of all.

As result, the only difference compared to the standard URS behavior – after handling the *agent becomes ready* event – the **VCB notifiable** call may be identified. And if such a call is identified, a VCB notification will be sent for it.

Important

A call is VCB notifiable if, the call is in the `DoNotSelect` mode (so it is automatically not routable); the call has the `notifyurl` property defined in its extensions; and, there are no pending VCB notifications for this call.

Reasoning: Every time some agent becomes ready and there is a VCB notifiable call waiting for that agent at the top of queue, URS will send a VCB notification for this call. If the corresponding customer is successfully dialed and the call's `DoNotSelect` mode is removed, then this call automatically becomes first in the queue and routable, so it will be answered when the next agent for this queue of calls becomes ready. This is expected to happen in approximately this queue's **Average Handling Time** (average agent's AHT/number of agents).

Important

If the **dial out hit rate** for the call is set to less than 100%, then URS might look for more VCB notifiable calls (up to 3). The **dial out rate** is set with the `DialOutSuccessRate` [rate of success in percent] strategy function . *0*, if used will be considered as *100%*. After finding the next VCB notifiable call, URS will use the provided value to consider as a probability that the customer answers the call. And URS will continue to search for more VCB notifiable calls with the probability % (100 - provided value). However, no more than 3 calls can be selected in total. For example, a DialOutSuccessRate of 70% will result in URS sending out one VCB notification with a probablity of 0.7%, two VCB notifications with a probability of 0.21% (03*0.7), and three VCB notifications with a probability of 0.09% (that is, 1.39% on an average).

Below is a listing of DialOutSuccessRates and their corresponding average number of VCB notifications that the `agent ready` event might generate:

DialOutSuccessRates	Avg. No. of VCB Notifications
100%	1
90%	1.11
80%	1.24
75%	1.3125
70%	1.39
60%	1.56
50%	1.75
40%	1.96
30%	2.19
20%	2.44
10%	2.71

Important

In the following sections (for simplicity) it is assumed that we have a 100% dial out hit rate (so that URS will look for only one VCB notifiable call each time).

Step 2: Improving VCB Notification Rate (since 8.1.400.23)

As can be seen, the functionality in Step 1 does (almost) not change from the standard URS behavior on processing the **agent becomes ready** event. This simplicity however might affect the rate of VCB notifications, which is about two times less than an optimal rate. So, while VCB customers are connected quickly with agents, the queue of VCB calls might be handled slowly.

The basic use case considered in this step - If only VCB calls are present in the system (that is, all calls that URS has are VCB ones and there are no other call types) and hit a dial-out rate of 100% - then the expectation is that the rate of VCB notifications will be the same as the rate of the **agent becomes ready** events (as there are no inbound calls occupying agents). This, however, does not happen within Step 1.

When two **agent becomes ready** events activate only one VCB notification:

- The **Agent 1 becomes ready** event triggers a VCB notification for Call 1.
- Call 1 is connected with a customer and the **DoNotSelect** mode for this call is cleared (so it is no longer VCB notifiable and is first in the queue).
- The **Agent 2 becomes ready** event triggers Call 1 to be sent to Agent 2, but it does not trigger a VCB notification as there are no VCB notifiable calls ahead of Call 1.
- Only when Agent 3 is ready will the next VCB notification be sent for the next VCB call.

To handle the above use case, the standard URS behavior on processing **the agent becomes ready** event was extended and goes beyond what is needed for simply routing calls. Specifically, URS does not stop when a routable call is found, but will try to continue exploring the queue of calls in search of a VCB notifiable one (if such a call is not already found). Previously, in URS, there was no such concept of *queue exploration* after a routable call was found. This behavior is limited to an extent so as to not affect URS performance in standard but not VCB-related cases.

The main idea behind this step is that, if a call is routed to an agent and the next call after it is a VCB notifiable one, then it is alright to send a VCB notification.

Implementation: URS continues to go through the queue of calls after a routable call is found until one of the below happens (first):

- A VCB notifiable call is found.
- Another routable call is found (meaning, when the next agent becomes ready, this call will be routed to that agent). All other VCB calls, even if they exist, have a lower priority and will not be routed. No more than 20 calls are checked (this is configurable, and is 20 by default).
- No more than 20 calls are checked (this is configurable, and is 20 by default).

Step 3: Further Improving (*increasing*) VCB Notification Rate (since 8.1.400.26)

While Step 2 is expected to help achieve a VCB notifications rate that is closer to the ideal rate, there are still use cases where this is not enough. As a result, extra logic is embedded from version

8.1.400.26.

The implementation described in Step 2 is updated as follows:

Use Case A

URS continues to go through its queue of calls after a routable call is found until one of the following happens first: - A VCB notifiable call is found. - Another routable call (but not a former VCB one) is found (no more than 20 calls is checked, this number is configurable).

Compared to Step 2, here URS continues to go through the queue of calls after a routable call is found until one of the following happens first:

- A VCB notifiable call is found
- Another routable call (but not a former VCB one) is found
- 20 calls are checked (this value is configurable and 20 by default)

The point here is taking into account the nature of another routable call. If it happens to be a former VCB call (a call that becomes routable due to a VCB notification that URS sent previously), then it should not prevent a search for other VCB calls. This use case can become significant on high call rates/VCB notification rates when more than one VCB call can become routable more or less simultaneously.

Use Case B

While the following use case could be an extremely rare one, it is still worth addressing.

If during the processing of an **agent becomes ready** event, no routable call was found, then arises a situation where there is an available agent without any call and it is expected that one more agent will become available soon (in queue AHT time). In such cases, URS will send two VCB notifications (meaning, a single **agent becomes ready** event can sometimes trigger two VCB notifications) one for the current agent and a second one for the agent who will become available next.

Since 8.1.400.26, this functionality is controlled with the **n10** parameter of the VCB option (see VCB Configuration) and is disabled by default.

Important

Before 8.1.400.40, this functionality was also disabled in a multi-URS environment (n4=1, see VCB Configuration).

Use Case C

URS has no control on how many VCB calls it gets and the queue of VCB calls could still grow (for example, if there are not enough agents). URS considers this as a normal situation if VCB calls and inbound calls are still treated fairly. A potential issue here is that a VCB notification does not block the agent and a lower priority inbound call will be routed to that agent (meaning, the low priority call will be routed ahead of higher priority VCB calls as they cannot be routed yet). In other words, low priority inbound calls might seep through the waiting queue of calls. To avoid this scenario, the following optional logic is added:

If for whatever reasons, URS gets a significant surplus of high priority VCB calls, then it tries to compensate for it. A surplus of high priority VCB calls occurs in the following condition:

- While going through a queue of calls upon processing **an agent becomes ready** event (in search of the first routable call)if URS finds that the head of the queue contains a large (>25, adjustable) amount of not-yet-tried VCB calls. In other words, there are a lot of high priority VCB calls and not a single high priority *non-VCB* call.

If URS detects such a situation, it stops looking for routable calls for this agent so no call is routed to the agent upon processing the **agent becomes ready** event. Referring to case B above, this means that two VCB notifications will automatically be sent for this **agent becomes ready** event. Also, the agent is marked as reserved for processing VCB calls only (or high priority inbounds calls). Such reservation continues while the surplus of high priority VCB calls exists.

Important

Before 8.1.400.40, this functionality was disabled in a multi-URS environment (n4=1, see VCB Configuration).

Step 4: Further Improving (*correcting*) VCB Notification Rate (8.1.400.26)

While looking for routable calls (applicable for all steps), URS checks not only the call state but the target (agent) state too. Specifically the following checks are carried out if the agent is not blocked completely:

- Is call state OK?
- Is agent valid?
- Is agent blocked?
- And for every agent’s DN: Is DN valid? Is DN blocked?

It is possible that a call is considered not routable only because of agent/DN blocking. In such cases, URS can potentially pass over a big number of calls as cannot be routed only because of agent blocking and reach a VCB notifiable call located well behind in the queue. Triggering a VCB notification in such a case could be incorrect – as another (or the same) agent becomes available the specific VCB call will not be routed to the agent (as there are a lot of other routable calls waiting).

In general, correctly distinguishing the above described case is a high CPU usage activity (because an agent can be partially blocked, that is, blocked for one call and not blocked for others). As a simplified alternative the following is done:

The cycle described in Step 1 is modified as follows: While going through the queue of calls waiting for a ready agent and looking for the first routable call, URS also looks for the first VCB notifiable call. However, as soon as such a call is not routable because of agent/DN blocking, then, since that moment the search for a VCB notifiable call is cancelled. In other words, blocked agents/DNs can trigger VCB notification only if a VCB call is the first call among all calls waiting for that particular agent).

Improvement described in Use Case B of Step 3 should also be disabled if agent is blocked (as this agent is not available and there is no justification to send the agent a second notification).

Improvement described in Use Case C of Step 3 should also be disabled if agent is blocked (as there is no way to check presence of routable calls in such a case).

Step 5: Further Improving (*extending*) VCB Notification Rate (8.1.400.36)

A multi-URS VCB environment is extended to allow using of an adjustable dial out hit rate. Previously dial out hit rates in case of multiple URSes was forced to always be 100%.

However, there are some limitations though (See Multiple URSes) and as a result it is advised to increase the priority of a VCB call after an outbound call was answered by the customer.

Step 6: Further Improving (*adjusting*) VCB Notification Rate (8.1.400.37)

The functionality implemented for Use Case B in Step 3 is disabled by default and explicitly controlled with the VCB option (n10=1, see VCB Configuration).

Step 7: Further Adjusting VCB Notifications (8.1.400.40)

From version 8.1.400.40, another requirement was being considered; prevent, as much as possible, lower priority inbound calls from seeping through the queue of not yet dialed/answered VCB calls. Basically, the only absolutely reliable way to achieve it is through VCB Type 1 (blocking/reserving agent after a VCB call). The below modifications were discussed in an attempt to address the issue within VCB Type 2 (notification based).

A way of addressing this requirement, is to provide a big enough pool of VCB calls already answered by customers, so that low priority inbound calls will not have chance to squeeze through the queue. Though seeping of low priority calls might only happen at the moment an agent is ready and there are no answered (ready for routing) VCB calls, a big pool of answered VCB calls would mean that these customers might wait for quite a long time before they can be connected with an agent - as the main reason for dialing them was to block inbound calls and not to facilitate a minimum waiting time after answering.

In effect, this requires URS to maintain a balance between *under-dialing* (some portion of inbound calls seeps through the queue of VCB calls) and *over-dialing* (once answered, customer waits too long before being actually connected with an agent) conditions. We can achieve this as follows:

a. To facilitate a situation where, URS always has VCB interactions that are already answered by customers, then URS must work in a slightly *over-dial* mode. The current default *optimal* rate (one notification per one agent ready event) will never be sufficient. The default rate of notifications URS will use now is 1.5 times higher than the rate at which agents become ready. One agent ready event on an average triggers 1.5 notifications (it is slightly less than the rate that resulted from a `DialOutSuccessRate` of 60). This can be controlled with the `n11` parameter of the VCB configuration option (see VCB Configuration). However, this parameter will be ignored if the `DialOutSuccessRate` function is used by a VCB solution.

b. The inherent *over-dialing* must be controlled and suspended/reduced when URS detects that customers of connected VCB calls are beginning to wait too long, as the criteria that URS uses is the time that connected VCB calls are already waiting for and also the number of already connected VCB calls. Specifically, when URS detects that a customer is waiting for more then one minute, it assumes a *over-dial* situation (controlled with the `n12` parameter of the `vcb` configuration option). The same is also true, if URS detects that there are already 5 answered VCB calls (controlled with the `n13` parameter of the `vcb` configuration option). In an *over-dial* situation, by default URS stops generating new VCB notifications while the *over-dial* situation continues.

With the `n14` parameter of the `vcb` configuration option, URS can be configured to still consider making VCB notifications but with a much smaller rate.

Important

This version changes the default URS VCB behavior. As in, it has a much higher dial out rate. This factor should be considered, when upgrading to 8.1.400.40. If there is a necessity to preserve the previous VCB behavior, then either the `n11` parameter of the `vcb` configuration option must be explicitly set to 100 or the `DialOutSuccessRate` function must be used. It is strongly advised not to mix both the approaches to control the dial out rate (`vcb:n11` parameter and `DialOutSuccessRate` function). If the `DialOutSuccessRate` function is going to be used, then it should be executed for every VCB call.

Note: The *over-dial* protection if enabled (with the `n12` and/or `n13` parameter) works irrespective of whether *over-dialing* results from the default dial out rate or using the `DialOutSuccessRate` function.

c. Using the jump priority functionality (the `n9` parameter of the VCB configuration option) might result in a sharp slow down in the VCB notifications rate. To handle this situation, when looking for VCB notifiable calls after a routable call is found (see the `n5` parameter of the `vcb` option in VCB Configuration), URS will not count the dialed but not answered VCB calls towards the `vcb:n5` parameter.

d. At times, there might be a few queues of VCB calls which are understaffed and result in extremely high wait times for processing every single call (20 to 30 minutes or even more). Such cases of very slow queues generally do not fit the notification model described here as it is practically not possible to find a right moment for sending the notification. The customer will almost always have to wait a significant amount of time after answering an outbound call. Cases like this are better addressed with the first type of VCB solution where the agent is actually allocated to an interaction from the moment a VCB call is dialed to a customer. As a compromise though, it is possible to configure a *hybrid* approach - in some cases URS might actually reserve an agent for a VCB call at the same moment that a dial out notification is sent and keeps the agent reserved until the customer answers. As this is effectively VCB Type 1 solution within a VCB Type 2 solution (see Preface section) it results in an instant connection of the call/customer with the agent, but the agent can still loose a part of his productive time waiting. This waiting might become bigger (up to the value in the n2 parameter of the `vcb` configuration option in VCB Configuration) if the customer does not answer and also in cases of a multi-URS VCB. This functionality is controlled with the `n15` parameter of the `vcb` configuration option in VCB Configuration. It specifies a minimum AHT for the queue where a VCB call is located and when this time is exceeded URS switches to the agent reserving mode (for calls from this queue only). By default, the value of this parameter is 0, meaning that this functionality is disabled.

In addition to checking queue AHT, URS might also check queue MHT (maximum handling time) and whether it exceeds the value of the `n15` parameter. Checking is done within the timeframe (seconds) defined in the `n16` parameter, which is 0 by default (meaning that the functionality is disabled). If within the defined timeframe, at least one call handling time for the queue exceeds the value specified in the `n15` parameter, URS will switch to the agent reserving mode (for calls from this queue only).

Important

Thoroughly consider all pros and cons if enabling this *slow queue* functionality. A major use case for this functionality is that, if there are only 2 or 3 agents to handle the calls from a queue and handling one call takes around an hour, then, maybe to facilitate an acceptable wait time (after customer answers an outbound call) it is alright to sacrifice one minute of the agent time. If so, then the `vcb:n15` parameter can be set to around 1200 (20 minutes) and URS begins to reserve the agent after the in-progress VCB call, if the next available agent is estimated to appear only after 1200 or more seconds.

Considering all the above, the process that URS follows (on a relatively low level) when going through a queue of calls waiting for an agent (when the agent becomes available) is as follows:

Here,

- `fresh_vcb_counter` is the number of VCB calls not yet dialed even once
- `vcb_main_pool_size` is the number of available slots in the pool of stored data for dialing VCB calls
- `vcb_extra_pool_size` is the number of available slots in the extra pool (one more VCB call might be stored for accelerated notification if no routing takes place or if the first and only VCB call from the main pool is not answered)
- `vcb_after_route_total` - number of calls in queue after a call was already routed to an agent
- `vcb_after_route_former_vcb` - number of calls in queue after a call was routed (that is, calls which are routable now and were former VCB calls, which had customers waiting)

Agent becomes ready.

- `fresh_vcb_counter = 0`
- `vcb_main_pool_size =` 1 to 3 (depending on how many notifications URS will generate on behalf of the agent)
- `vcb_extra_pool_size = 1`

If current ratio for VCB notifications is different from 100% (1:1), `vcb_extra_pool_size` = 0 (if `vcb_main_pool_size` might be more than 1, then the extra pool is not used).

If agent ID, place, or DN is blocked (or was used in recent past), `vcb_extra_pool_size` = 0 (see Step 4 above).

Loop 1:

For every call waiting for this agent (in priority/best fit/timing order):

- if call is routable, try to route it.
    - if success (call is routed), exit from Loop 1.
    - if failure (call is not routed) because agent/place/DN is blocked, `vcb_main_pool_size` = 0 (see Step 4 above).
- if call is VCB notifiable:
    - if call was not yet dialed, `fresh_vcb_counter++`
    - if `fresh_vcb_counter > vcb:n8`, exit from Loop 1 (if at this point, both pools of VCB calls are completed, `vcb_main_pool_size` and `vcb_extra_pool_size` are 0)
    - if `vcb_main_pool_size > 0`, store the call in the main VCB pool, and `vcb_main_pool_size--` else, if `vcb_extra_pool_size > 0`, store the call in the extra VCB pool and `vcb_extra_pool_size--`

At this point, if a call is routed and there are not enough stored VCB calls (`vcb_main_pool_size > 0` or `vcb_extra_pool_size > 0`), then Loop 1 is exited and Loop 2 comes into effect.

Loop 2:

- `vcb_after_route_total = 0`.
- `vcb_after_route_former_vcb = 0`.

For every call waiting for this agent, after the just routed call (in priority/best fit/timing order):

- if call is not a dialed but not answered VCB call, `vcb_after_route_total++`
- if `vcb_after_route_total > vcb:n5`, exit Loop 2
- if call is routable:
    - if call is not a former VCB call (that is, call is a regular inbound call) then exit Loop 2
    - if call is a former VCB call, then `vcb_after_route_former_vcb++`
    - if `vcb_after_route_former_vcb > vcb:n13`, then enter the *overdial* mode
    - if the customer for this call has already been waiting for a considerable time, then enter the `overdial` mode
    - if overdial mode,
        - with probability of 100 - `vcb:n14`, then empty pools of collected VCB calls (if any) and exit Loop 2
        - if pool of collected VCB calls is not empty, then leave one call in the queue, and exit Loop 2
        - `vcb_main_pool_size = 1` (with probability `vcb:n14`, allow room just for one VCB call)

        - `vcb_extra_pool_size= 0`
    - if VCB call is notifiable:
        - if `vcb_main_pool_size > 0`, store the call in the main VCB pool and `vcb_main_pool_size--`
        - else, if `vcb_extra_pool_size > 0`, store the call in the extra VCB pool and `vcb_extra_pool_size--`
        - if `vcb_main_pool_size = 0` and `vcb_extra_pool_size = 0`, then exit Loop 2 (there are enough stored VCB calls)

At this point, both Loop 1 and Loop 2 are exited.

- if there are calls in the main VCB pool, send VCB notifications for each one of the calls
- if there are calls in the extra pool,
    - if no call was routed and `vcb:n10` is 1, then send a VCB notification for this call too.
    - otherwise store information about this call in call1 from the main VCB pool (could be only one call there too) - if call1 customer does not answer, the VCB notification will be sent for this stored call

Additional VCB features in this version are:

- Option `vcb` (see VCB Configuration) can be set at the call level with the `SetCallOption['vcb', '...']` function. Not every parameter of this option has a meaning at the individual call level and accordingly only a few parameters of this option can be set at the call level. Specifically, the `n1`, `n2`, `n6`, `n9`, `n17` parameters can be set the call level. Use values -1 for all other parameters. If the value of a parameter is set to -1, its value will be taken from vcb option at the URS level.
- The `vcb` option is dynamic. It does not require a URS restart if modified.
- Every parameter of the `vcb` option get its own name and can be specified on its own (see VCB Configuration).
- The default value of the `n8` parameter of the `vcb` option is 25 (previously, it was 50).
- The default value of the `n1` parameter of the `vcb` option is 60 (previously, it was 30).