Universal Routing Voice Call Back

Last modified on Dec 07, 2016

`

Preface

Universal Routing Server (URS) manages two types of Voice Call Back (VCB) solutions:

1. Preemptive Agent Reservation

   In this process, an agent is reserved in advance before a VCB call is made to reach the customer. This call flow is not that different from inbound interaction routing: An agent is selected for a call, this could be a "VCB call", and the customer and the agent are subsequently connected. The connecting phase could be more complex for VCB calls, as the connection entails first dialing the customer as compared to regular routing.

   - Pro: An advantage of this approach is that the customer is guaranteed not to wait after answering the VCB call, as the associated agent is already allocated.

   - Cons: Agents can lose time – In this scenario, Agents are made to wait for the interaction to arrive during the time the customer is being contacted. If in case the customer does not answer, the agent wait time is lost. This in turn will affect Agent Productivity Statistics.

2. Dialing Notifications

   The second type of VCB is when an agent is not assigned to the call, until the customer calls back. In this case, the customer is contacted in advance, before agent is allocated to the VCB call. The allocation will only be done after the customer answers the call.

   - Pro: Agent will not lose time as they will not be waiting for the call to be answered.

   - Cons: In an under staffed environment, situation could raise where we would not be able to route the call to an agent right away, Customer may need to wait in queue.

It is generally assumed that URS uses this second type of "dialing notifications" approach for VCB, notifications that URS can generate under certain conditions.

This white paper focuses on URS "Dialing Notifications".

The purpose of URS VCB notifications is to facilitate:

- The maximum possible rate of the dialing of outbound calls.
- The minimum possible customer waiting time after answering the call (ideally the customer will be connected with first agent, who becomes available).

This means that URS VCB intends to achieve the following rate of dialing outbound calls (called *optimal* in this document): **rate that agents become available minus the rate of inbound calls**

`

Standard URS Behavior (no VCB)

In a call surplus scenario:

- Agent becomes ready.
- An Event from Stat Server is received.
- URS builds a queue of all calls waiting for this agent based on priority, waiting time, etc.
- 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).
- Once (if) such a call is found, it is routed to that agent.
- Processing of the "agent becomes ready" event is complete.

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

In the following description, it is assumed that we have a 100% dial-out hit rate.

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 also 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 standard URS behavior – after handling "agent becomes ready"– the **VCB notifiable** call may be identified. And if such call is identified, a VCB will be sent for it.

Note - A call is **VCB notifiable,** *if the call is in DoNotSelect mode (so it is automatically not **routable**); the call has **notifyurl** defined in its extensions; and there are no pended VCB notification for this call.*

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).

Step 2. Improving VCB Notification Rate (Since 8.1.400.23)

As can be seen, the functionality on 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 the "optimal" one. So, while VCB customers are connected quickly with agents, the queue of VCB calls might be handled slowly.

Basic use case considered on this step – Only if VCB calls are present in system 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

`

“agent becomes ready” events (as there are not any inbound calls occupying some agents). This, however, does not happen within Step 1.

Details when two “agent becomes ready” events activate only one VCB notification:

- Agent1 becomes ready event triggers a VCB notification for call1.
- Call1 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).
- Agent2 becomes ready event triggers call1 to be sent to the Agent2, but it does not trigger a VCB notification as there are no VCB notifiable calls ahead of call1.
- Only when an Agent3 becomes ready will the next VCB notification be sent for the next VCB call.

To handle this 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 call is not already found of course). Previously in URS, there was no such “queue exploration” after a **routable** call was found. The behavior is limited in depth so as to not affect URS performance in standard but not VCB-related cases.

Main idea behind this step: If a call is routed to an agent and the **next** call after it is a **VCB notifiable** one, then it is Ok to send a VCB notification.

Implementation: URS continues to go through the queue of calls after a **routable** call is found until whatever below happen first:

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

Step 3. Continuing improving (increasing) VCB notification rate (since 8.1.400.26)

While Step 2 is expected to make the rate of VCB notifications closer to the ideal one, there are still use cases when it is not enough. As result the following extra logic is embedded.

Described implementation for Step 2 updated in following way:

A. URS continues to go through its queue of calls after a **routable** call is found until one of the following happen first:
   - **VCB notifiable** call is found.
   - another **routable** call (but not former VCB one) is found.
   - no more 20 calls is checked (configurable, 20 by default).

   Point here is taking into account the nature of another **routable** call. If it happens to be a former VCB call (call that becomes routable due to VCB notification that URS sent previously), then it should not prevent searching for other VCB calls. This use case can become significant on high

`

call rates/VCB notifications rates when more than one VCB call can become routable more or less simultaneously.

B. While the following use case is somewhat artificial, it is still worth addressing.

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

C. URS has no control on how many VCB calls it gets and the queue of VCB calls can still grow (for example, if there are not enough agents, etc.). URS considers this as a normal situation if VCB calls and inbound calls are still treated fairly. A potential issue here is that VCB notification does not block the agent and some lower priority inbound call will be routed to that agent (= the call will go ahead of higher priority VCB calls as VCB calls cannot be routed yet). In other words, low priority inbound calls might seep through the waiting queue of calls. As result, the following optional logic is added:
   - If for whatever reasons URS get a significant surplus of high priority VCB calls, then it tries to compensate for it. A surplus of high priority VCB calls is comprised of the following condition:

     Calls are going on through the queue upon processing an “agent becomes ready” event (in search of first **routable** call) and URS finds that the head of queue contains a large (>50, 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 "not VCB" call.

     If URS detects such 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, for that matter). Such reservation continues while the surplus of high priority VCB calls exists.
     Note. This last C case is always disabled in a multi URS case (n4=1, see below).

URS VCB Related Logging

While going through a queue of calls in search of the first **VCB notifiable** one for which a notification can be sent, URS logs a short disposition about every checked **VCB notifiable** call. The corresponding logging message has form:

```
_M_I_connid [10:21] call VCB states N (d1:d2:d3)
```
For example: `_M_I_036f02849e162002 [10:21] call VCB states 1 (0:0:0)`

A message is printed for every checked VCB call (one in *DoNotSelect* mode and having notifyurl defined):

N = **0** if for this call a VCB notification was sent recently (within n2 seconds, see VCB Configuration section).

`

N = **1** if the call cannot be routed right now (even if there was no *DoNotSelectMode*) due to call status.
N = **2** if VCB notification was already done on behalf of the currently processed target (agent) recently (within n3 seconds , see VCB Configuration section) .
N = **3** if the call is good for VCB notification.
N = **5** if the call is already scheduled (but VCB notification is not yet sent) and it is not first among such calls.
N = **6** if first call that already has been scheduled (but not yet sent VCB notification). Used as an extra/alternative VCB call.
N = **7** n7 parameter (see VCB Configuration section) is set to 0 and URS checks VCB call extra condition and it fails due to Threshold/ReadyConditions.
N = **8** n7 parameter (see VCB Configuration section) is set to 0 and URS checks VCB call extra condition and it fails due to target/agent validness verification.
N = **9** n7 parameter (see VCB Configuration section) is set to 0 and URS checks VCB call extra condition and it fails due to target/agent DN validness verification.
N = **10** next good VCB call after the one already selected. Used as an extra VCB call.

When URS selects a VCB call for notification and starts the notification process, it logs a message similar to the following:

If the notification is actually sent:

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web
notification <http://10.179.117.52:8080/genesys/1/ors/scxml/... > sent
(hints: 0 0 1464877932 60 0 2ac076be1f70)
```

If the notification is delayed due to high EWT or the necessity to send a reserving request (n1 – is 1 if high EWT current notification delayed due to high EWT; n2=1 if reserving request is sent; n3 – timestamp when VCB notification for this call can be repeated; n4 –in case of high EWT – for how long notification is delayed, pvq – points to internal queue containing the agent):

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web
notification <http://10.179.117.52:8080/genesys/1/ors/scxml/... > delayed
(hints: n1 n2 1464877932 60 0 2ac076be1f70)
```

If it is an attempt to send one more notification for the call, which has delayed notification, then notification will not be sent and URS places in log:

n1 – is 1 if current notification delayed due to high EWT

n2 – is 1 if current notification is delayed due to an agent reserving request

n3 – timestamp when VCB notification for this call can be repeated

n4 – number of seconds left to this moment of time

pvq – points to internal queue containing the agent

```
16:31:12.006_A_E_036f02849e162002 [0E:22] web notification failed
(hints: n1, n2, n3, n4, 0, pvq)
```

If a notification was sent but resulted in no response and URS is going to send another notification, the following is printed regarding an “expired” notification:

n3 – timestamp when VCB notification for this call can be repeated.

n4 – number of seconds left to this moment of time.

`

timed out is used if the timestamp is actually expired, **cleard** if doing it before the timestamp expired.

```
16:31:12.006_A_I_036f02849e162002 [0E:22] web notification cleared|timed
out (hints: 0, 0, n3, n4, 0, 0)
```

If any notifications sent for this call is aborted (due to high number of previously sent unanswered notifications (URS place number of unanswered notifications in hints):

```
16:31:12.006_A_E_036f02849e162002 [0E:22] web notification aborted
(hints: 0 0 0 0 n 0)
```

When URS sends a VCB notification because of an update of some target (agent) this target is blocked for some time to prevent/provoking the sending of multiple notifications (see n3 in see VCB Configuration section). The appropriate logging message looks like:

```
16:31:12.006_M_I_036f02849e162002 [0E:25] SO(2ac076be44e8 13 2)
ten=Resources name=1805@STAT01.A: VCB notification timeout set: 1464877932
(+60)
```

URS can also clear such target blocking of VCB notifications if needed with the appropriate message:

```
16:30:21.745_M_I_0000000000000000 [0E:25] SO(2ac076be44e8 13 2)
ten=Resources name=1805@STAT01.A: VCB notification timeout clear: 0 (+0)
```

Voice Callback Configuration

URS has the option **VCB** (not dynamic)**,** which controls different aspects of VCB notification functionality in URS. The value of this option has the format of a sequence of numbers:

`n1:n2:n3:n4:n5:n6:n7:n8:n9` and by default is `30:90:90:0:20:0:1:50:0:0`.
The meaning of these numbers are as follows:

n1 (**EWT threshold**, by default 30). When an “agent becomes ready” event triggers a VCB notification, URS can send the notification immediately or with some delay. If the expected time when the next agent will become ready (queue AHT time) is high, then URS can decide to delay notification and not alert the customer too early**. Specifically, if this time is more than n1 seconds, then URS delays notification by n1/2 seconds.** AHT time here is counted for URS’s queue (that the VCB call is in).

RvqData[…, RVQ_DATA_AHT]. It is the same value as returned by URS function RvqData[internal_queue_id, RVQ_DATA_AHT].

n2 (**block call**, by default 90). When VCB notification for some call is sent, URS might not know what happened with this notification. Was it processed somehow, what is result of this processing (there is no formal feedback event)? As a result, the VCB call might remain in VCB notifiable state and trigger another notification on behalf of another agent, etc. To prevent a single VCB call from “hogging” all agents once a VCB notification is sent for some call, the call is blocked from sending any other VCB

`

notification for the next n2 seconds. If, after expiration of this interval, the call is not yet routed and is still in VCB notifiable state, the next VCB notification can be applied to this call.

n3 (**block agent**, by default 90). Effectively the same, but in the context of agents. If an agent becomes ready and triggers some VCB notification, this agent can remain ready (no other calls sent to him) and, as a result, trigger other VCB notifications. To prevent this from happening (not to generate too many notifications), this agent is blocked from triggering any other VCB notifications (for other calls) for n3 seconds.

n4 (**multi URS VCB**, by default 0). Can be 0 or 1 and indicators whether any other URS exists that can route call to the same agents. When VCB notification for some call is made, it is expected that this VCB call is first in the queue. Every URS, however, knows only his own queue meaning that other URSes can have higher priority calls and, in such cases, no VCB notifications should be made. In other words, this is flag indicates whether or not there are multiple URSs. If multiple URS mode is set for VCB notifications, then it also must be set for regular routing – in other words, URS also must have option **agent_reservation** on (if not, this n4 value will be ignored).

If n4=1 then before making VCB notification (or before routing) URS will try to **explicitly** reserve the agent triggering VCB notifications with priority data taken from the VCB call (or the call that is routed). VCB notifications will be distributed only if and by those URSs who get confirmation on this reserving request. The reserving request is made on behalf of the agent DBID and, as a result, does not interfere with routing reservation. It also means that when a call is selected and routed, it might result in two reserving requests to T-Server – one VCB-related and another regular routing related. *Note. If set, this flag works even if URS does not participate in VCB. It just causes URS to signal about the call when routing the call. And VCB URSes (if they happen to exist) will use them to adjust the sending of VCB notifications.*

n5 (**after route queue max**, by default 20). See Step 2 and Step 3. Controls how far URS goes through queues of calls after a routable call is found.

n6 (**max notifications**, recommended to set to 5 but for compatibility, by default is 0). Can be used for tracking and eliminating dead VCB calls. Lost (stuck) VCB calls are those that do not exist in the module accepting VCB notifications and, as a result, VCB notifications for such calls are just ignored. That can easily result in a small amount of dead VCB calls that can clog distribution of VCB notifications completely. All VCB notifications (or biggest part of them) will be for such dead calls. If n6 is not 0, then for every VCB call, URS counts the number of unanswered VCB notification in row. The counter is cleared only if URS get some external message (does not matter what about) for this call (web request message from ORS, etc.) If the counter manages to reach n6, then such call is temporarily removed from the pull of VCB calls for 20 minutes. If, after expiration of this time, the counter is still not cleared, then one more VCB notification for this call is allowed and, if it still results in no action, the processing of this VCB call is terminated inside URS.

n7 (**use any agent**, by default 1). Allows using any ready agent for triggering VCB notification. A routing strategy (including those processing VCB calls) can impose many different extra conditions on agents that the call can be routed to (different thresholds, etc). If n7=0, then URS checks all those conditions for VCB calls to become eligible for VCB notifications. Generally, it is not possible to predict when the customer will answer the call and some other agent will become ready – whether or not all those conditions be satisfied.

n8 (**check queue**, by default 50). See Step 3, case C. Controls the definition of "*Surplus of high priority VCB calls*" case.

n9 (**jump priority**, by default 0). Allows to automatically boost call priority by n9 when VCB calls becomes routable (customer answers the call).

n10 (**one blocker**, by default 0). See Step 4. If set to 1 then "not VCB call" can block VCB notification just once.

`

Multiple URSes.

Using multiple URSes means that agent reservation must be activated (even without the context of VCB). URS option **agent_reservation** should be set to **true**, **generic** or **implicit** on all URSes. Any involved URSes and T-Servers also must be configured to support the selected agent_reservation method.

Additionally (VCB Specific):

On **all** URSes (even those not involved in VCB but just routing calls on the same agents as VCB URSes), option **vcb** must have n4 value set to 1 (see section **VCB Configuration**).

As VCB notification cannot use implicit agent reservation, all URSes (even those not involved in VCB but just routing calls on the same agents as VCB URSes) must be configured to be able to perform explicit agent reservation (regular or centralized). Even if option **agent_reservation** is set to **implicit**, these URSs still need to be able performs explicit agent reservation (have a connection to centralized or all involved T-Servers) if needed.

Use of multiple URSs disables the functionality described in Step 3 part C. For VCB notifications URSs will assume a hit rate of 100%.

*If multiple URSes are used for VCB, but not configured as such (n4 in **VCB** is 0), the VCB functionality still will work. However, the functionality can potentially result in an over-dial of VCB calls (in effect, this is the other way to set a smaller hit rate)*.

VCB Notification Structure

As previously mentioned, a VCB notifiable call must have in its extensions the key **notifyurl**. This key, together with few other optional extensions, controls the location and format for sending VCB notifications.

Extra extensions keys are **notifybody** and **notifyenc**.

The sending VCB notification logic is as follows:

For an ORS instance:

- If **notifyurl** starts with "ors://" then the entire **notifyurl** should be in format:

  `ors://orsname/scxml/session/orssession/event/eventname[?params]`

URS will try to send the event to the specified ORS and session:

- If **notifyurl** starts with "ors:" then the entire **notifyurl** should be in format:

  `ors:whatever/event/eventname[?params]`

URS will try to send the event to the specified ORS node and session associated with this VCB call.

In both cases, optional params and also notifybody can be provided.

- `Params` should have the format of an urlencoded string of parameters.
- `notifybody` could be either an urlencoded string of parameters or JSON string.

`

Their values, if provided, will be decoded first – any fragment in square brackets will be replaced with its meaning value:

`[udata]` – entire call attached data (& separated)
`[udata.*]` – entire call attached data (, separated)
`[udata.key]` – value of corresponding attached data
`[udataj]` – entire call attached data as JSON string
`[ext]` – all call extensions (& separated)
`[ext.*]` – all call extensions (, separated)
`[ext.key]` – value of corresponding extension key
`[extj]` – all call extensions as JSON string
`[orssession]` – ORS session ID if call has associated ORS session ID
`[ors]` – host:port of ORS node associated with this VCB call
`[call.connid]` – Connection ID of this VCB call
`[call.uuid]` – UUID of this VCB call

For an URS instance:

- If **notifyurl** starts with "urs://" then entire **notifyurl** should be in format:

  `urs://ursname/message`

URS will try to send the command to the specified URS.

- If **notifyurl** starts with "urs:" then entire **notifyurl** should be in format:

  `urs:message`

URS will try to send the command to itself.

In both cases, the command is effectively invoking the RequestRouter function (updated description in the *Supplement to the Universal Routing 8.1 Reference manual*):

- `RequestRouter[ursname, message, notifybody, '', ''].`

  Both `message` and `notifybody` preliminary will be decoded just as described above.

For a generic HTTP server:

In all other cases, URS will try to send a VCB notification as REST HTTP message.

- **notifyurl** to be valid url.
- **notifybody** – If specified, URS will use POST instead of GET message and using notifybody as content of this POST message.
- **notifyenc** – Used only if notifybody is present and will be used as the content of Content-Type header of the generated HTTP message.

Both notifyurl and notifybody preliminary will be decoded as described above.

`

URS VCB Sample

Below is sample implementation of URS VCB. For simplicity and compactness, the sample uses a pure URS-based approach. All VCB components (sending and processing VCB notifications) are written as URS strategies.

To enable VCB functionality, the call processed by URS must be marked as VCB-enabled. This is achieved by attached data ECECUTION_MODE set to VCB. That attached data will not affect the standard processing of an inbound call. However, if an inbound call happens to be abandoned before routing, then URS will not drop it, but will continue the strategy as if nothing happened and will try to allocate this call to an agent and connect the customer and the agent.

Just setting this attached data is already enough to facilitate basic VCB **of type 1** for any existing solution – no strategy changes are required. The needed outbound call to the customer will be generated and, after answering, the customer will be connected with the selected agent. This type 1 VCB (when an agent is first selected/reserved after some call and, after that, an attempt to connect customer and agent is started) can be further adjusted if needed with extra attached data keys:

- VCB_CONTACT – Where to dial the customer (by default ANI is used).
- VCB_STRATEGY - If the logic for processing an inbound call **after/if** it was abandoned and turned into VCB, if the logic is different from original is required.

3. For implementing/enabling notification-based VCB (VCB of type 2) a few more steps are required (compared with the agent reserving VCB case).

   a. Attached data **EXECUTION_MODE=VCB** - That is the common step. It is required to make URS keep the call after the caller hangs up.

   b. Execute function **DoNotSelectCall[true]** – It will prevent URS attempts to select an agent for this call (as an agent is not supposed to be allocated until customer answers the outbound call).

   c. Execute function like ExtensionUpdate['notifyurl', 'urs:urs/call/[call.connid]/start?strategy=**ConnectToCustomer**'].

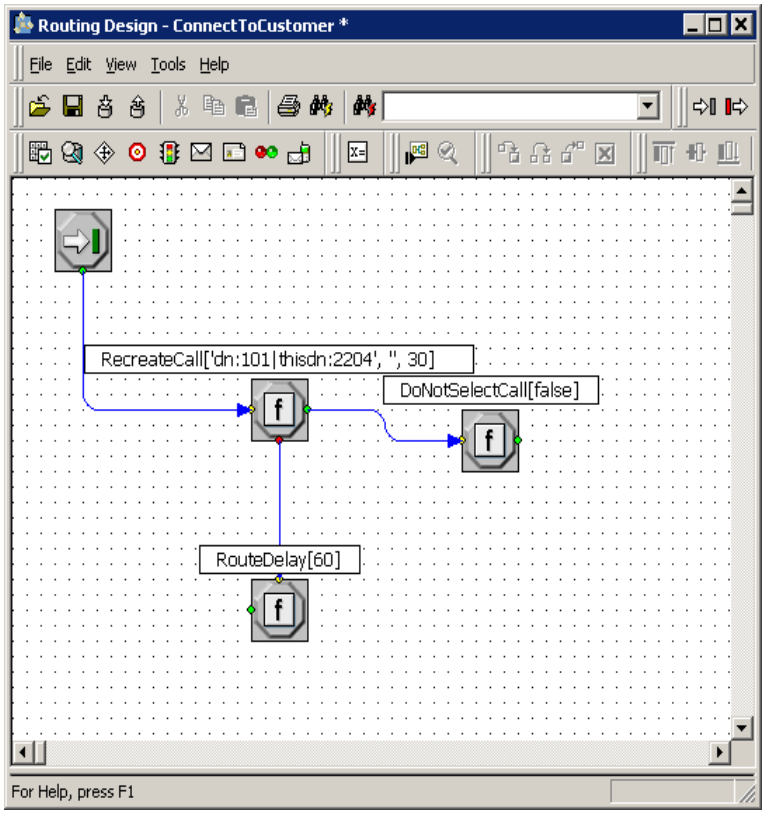
   Together with step b, it will enable URS to activate logic in strategy **ConnectToCustomer** when URS decides that it is time to dial the customer presented with this VCB call. The name of the strategy can be anything. In this sample, the name is **ConnectToCustomer**.

   d. Write the above mentioned **ConnectToCustomer** strategy. The strategy is simple and can be directly encoded into notifyurl itself such as by using the following notifyurl instead the one provided on step c:

   urs:urs/call/[call.connid]/start?source=if(Error(RecreateCall(ANI(), '', 30))) RouteDelay(300); else DoNotSelectCall(false);

Or this logic can be included into a separate strategy and referred to by name (see **ConnectToCustomer** strategy below):

`



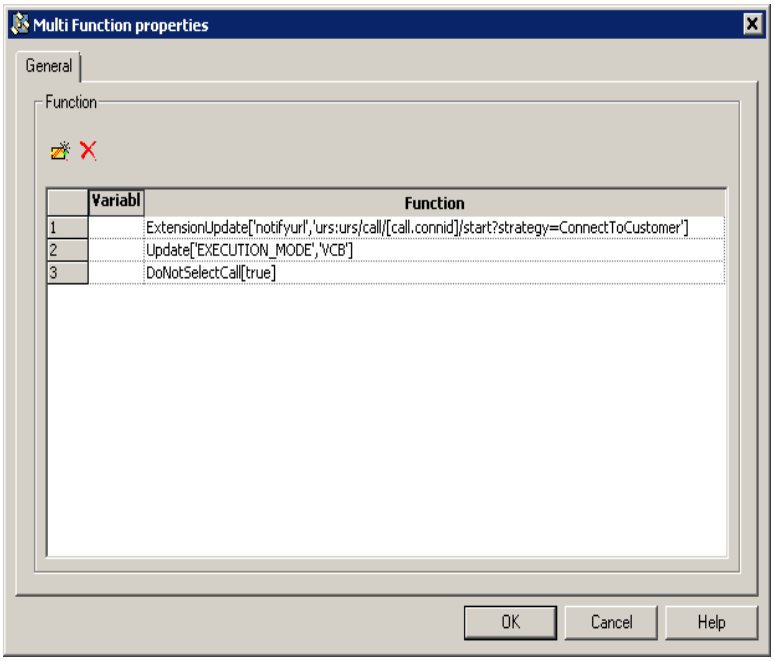
- Function **RecreateCall** just tries to connect to the customer and, if it succeeds, returns Ok and an error otherwise. In the case of the success DoNotSelectCall flag canceled, nothing prevents this call from being routed to next available agent. In the case of failure, we pause for some time before the next attempt to dial up the customer.

- Function **RouteDelay**[time] will make a call unnotifiable (In addition to being unroutable) for the provided time and as such, URS will not try to issue another notification.

`

Summary

In recap, activation of notification-based VCB consists of the following set of functions executed before the customer accepts the VCB offer:



`