Voice Callback and URS

VCB Sample

5/5/2025

VCB Sample

This section shows a sample VCB implementation in URS. It is simple and short using a completely 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 setting the EXECUTION_MODE key of the attached data to VCB. The attached data will not affect standard processing of inbound calls. However, if an inbound call happens to be abandoned before routing then URS will not drop it but will continue executing the strategy and will try to allocate the call to an available agent and connect the customer and the agent.

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

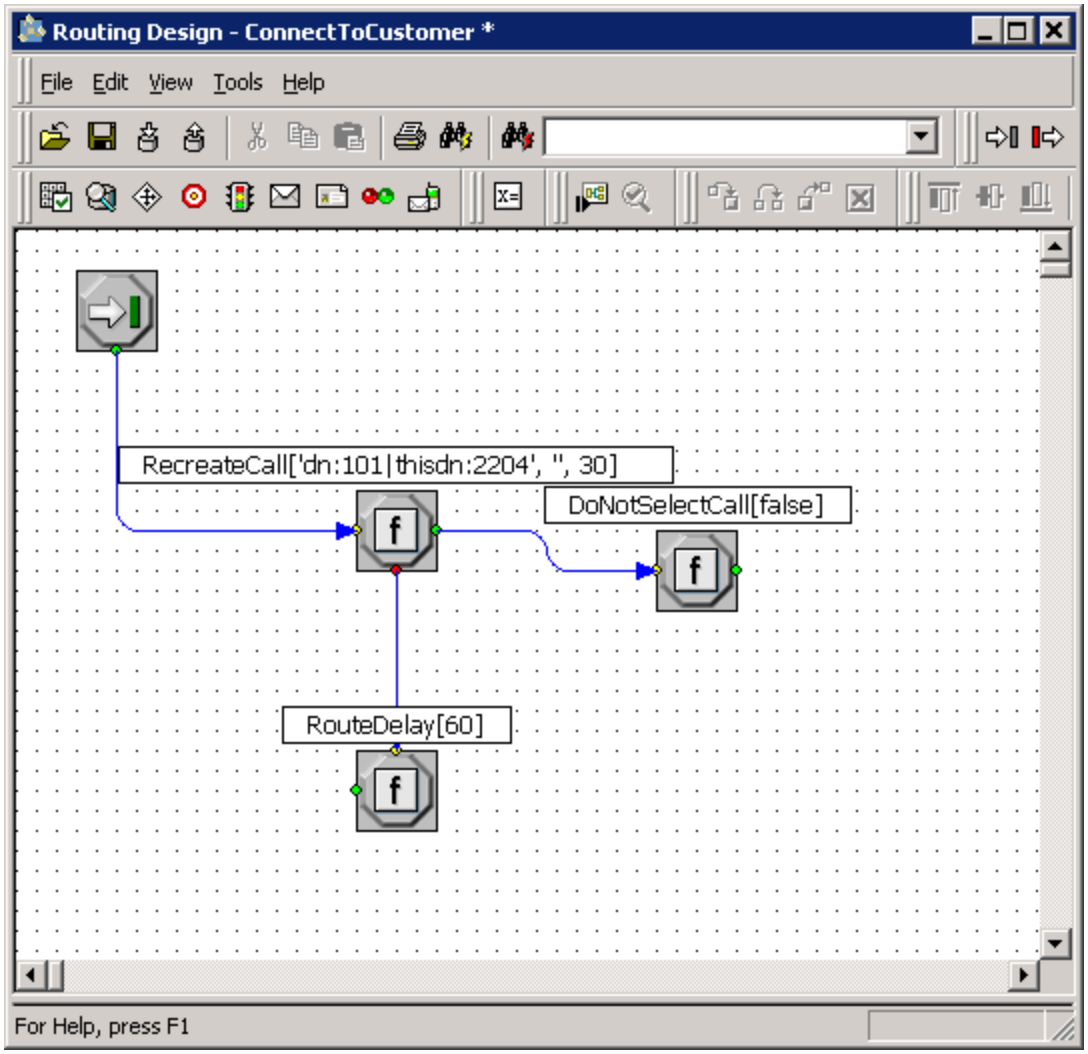
- `VCB_CONTACT` – where to dial customer (by default ANI is used)
- `VCB_STRATEGY` - if logic for processing inbound call after/if it was abandoned and turned into a VCB call must be different from the original logic and so on.

For implementing/enabling notification based VCB (VCB of type 2) a few additional steps must be performed (compared with the agent reserving VCB case).

- a) attached data EXECUTION_MODE=VCB - Common step required to make URS keep the call after the caller hangs up.
- b) execute function DoNotSelectCall[true] – To prevent URS from attempts to select an agent for this call (as agent is not supposed to be allocated until customer answers the outbound call).
- c) execute function ExtensionUpdate (for example, ExtensionUpdate['notifyurl', 'urs:urs/call/[call.connid]/start?strategy=ConnectToCustomer']) - Together with step b this will enable URS to activate logic in the `ConnectToCustomer` strategy when URS decides that it is time to dial the customer presented with this VCB call (name of the strategy can be anything; in this sample it is named `ConnectToCustomer`).

Note: The above will result in the following VCB notification to be distributed, `urs:urs/call/[call.connid]/start?strategy=ConnectToCustomer`. In other words, this means URS gets a *web* request to start executing the logic defined in the provided strategy in the context of the interaction with the provided `ConnID`. Execution of the logic is started in a parallel thread (see the description of the `start` URS WEB API method) and will not interfere with the main logic to be executed for this interaction.

- d) Write the above mentioned `ConnectToCustomer` strategy. The strategy is simple and can be directly encoded into `notifyurl` itself, for instance, by using the following `notifyurl` instead the one provided in step c: `urs:urs/call/[call.connid]/start?source=RecreateCall(ANI(), , 30); if(Failed()) RouteDelay(300); else DoNotSelectCall(false);`. Or this logic can be included into a separate strategy and referred to by name (see the `ConnectToCustomer` strategy below):

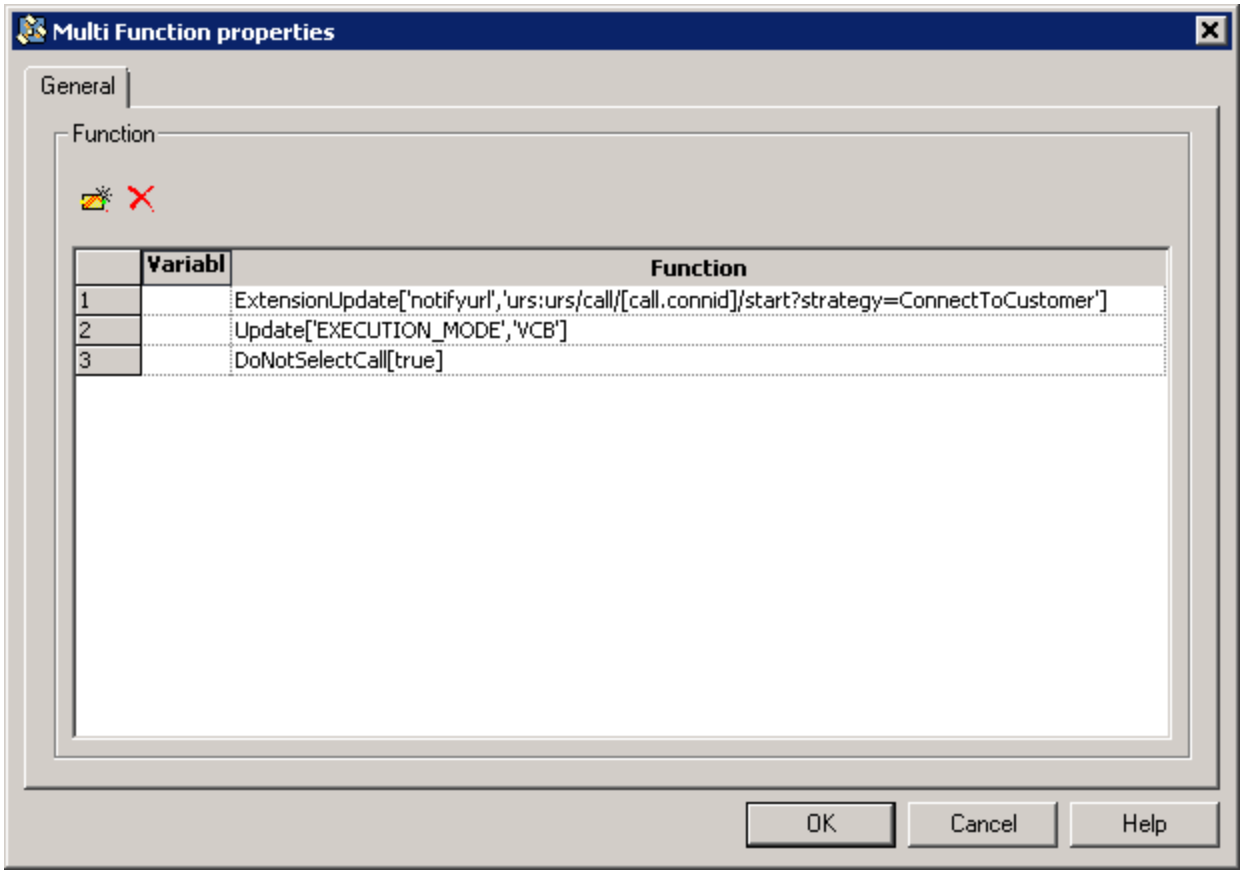


The `RecreateCall[dn, extensions, timeout]` function tries to establish connection with customer and if it succeeds, returns OK or otherwise an error (in effect, it converts a *virtual VCB call* into a real one). Its parameters are: where to dial the call, which extensions to specify in dialing request to T-Server, and how long to wait for customer to pick up the phone.

In case of success, the DoNotSelectCall flag is canceled and nothing prevents this call anymore from being routed to the next available agent. In case of failure, we pause for some time before the next attempt to dial the customer.

The `RouteDelay[time]` function 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.

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



Important

The `DoNotSelectCall[true]` function must be executed just before the customer accepts the VCB offer (and if he does not, then it should be canceled if it was already raised).