

### Description

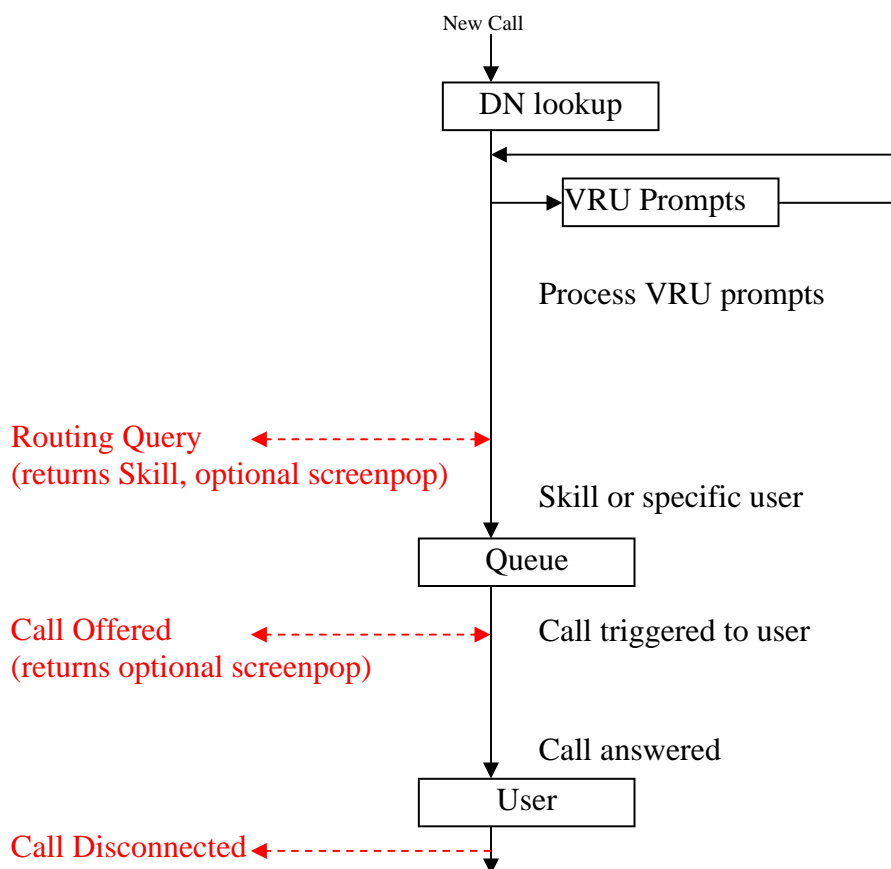
The OnState Call Event Interface provides call events for integration with CRM systems or other business applications. Events are sent to a customer webservice as defined here.

The system currently supports three call events.

- The RoutingQuery event is sent by a step in the call routing process. It sends call data and expects a routing Skill as return. The routing step includes a default skill to be used if no valid Skill is returned.
- The CallOffered event is sent when a user is selected to receive the call. It includes all call data and an identifier for the selected user.
- The CallDisconnected event is sent when the call terminates. It provides answer and disconnect times, so that duration information is available for the business application. This event is sent for both answered and abandoned calls.

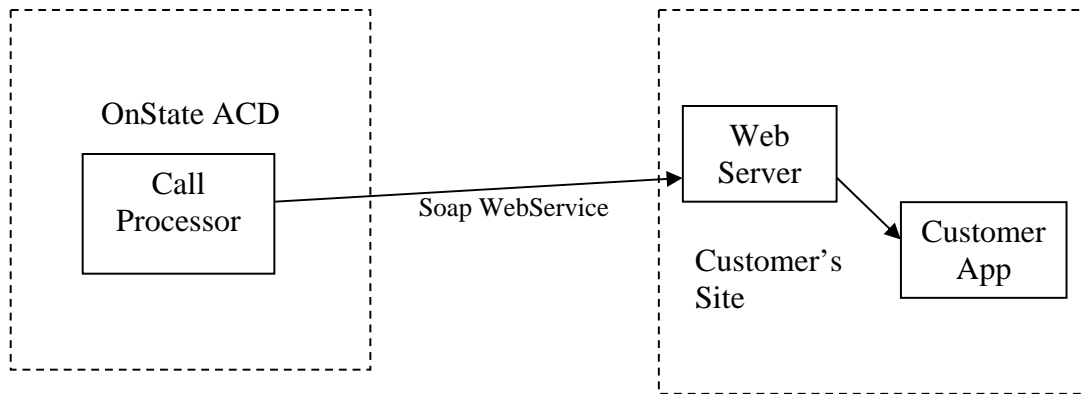
The RoutingQuery is configured by a customer through the OnState Supervisor. The CallOffered and CallDisconnected events are configured by OnState.

### Call Processing Model



### Message delivery

Messages are sent to the customer application from the OnState call processor. The receiving address is configured for the OnState system and is used for all messages.



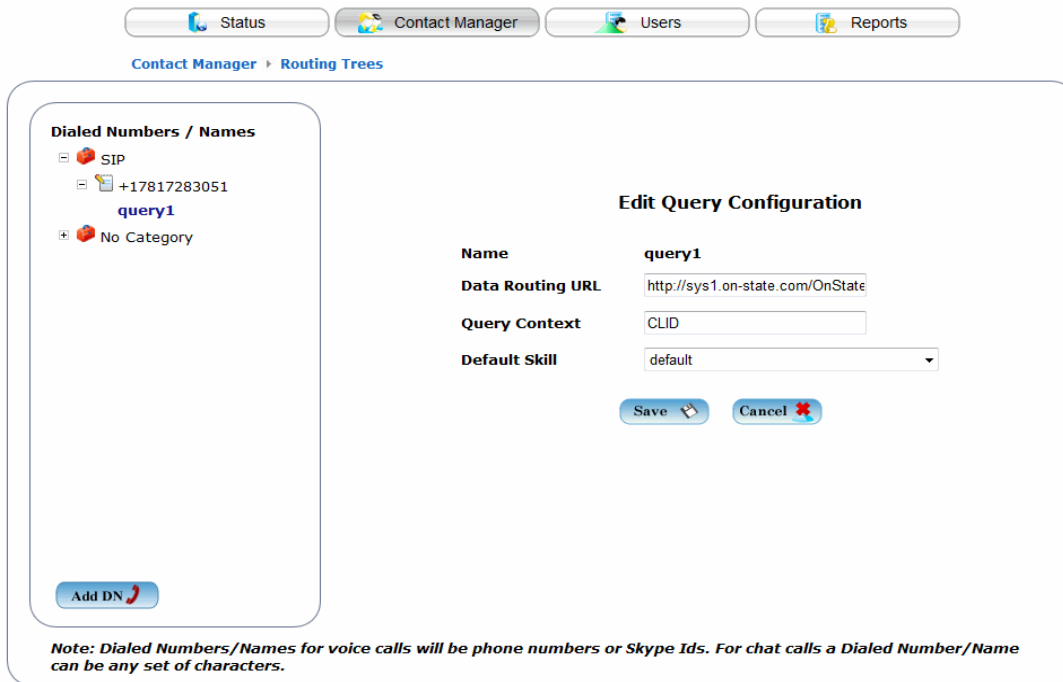
Webservice Interface

All messages are wrapped by a Soap wrapper and sent as the body of an HTTP request.

## Call Events

### RoutingQuery Event

This event is sent as a result of a RoutingQuery step in the routing tree. The following figure shows the configuration screen for a RoutingQuery.



The screenshot shows the 'Edit Query Configuration' screen in the OnState Call Event Interface. At the top, there are four navigation buttons: 'Status', 'Contact Manager', 'Users', and 'Reports'. Below these, the 'Contact Manager' section is active, showing a tree view of 'Dialed Numbers / Names' with categories like 'SIP', '+17817283051', 'query1', and 'No Category'. The 'Edit Query Configuration' form is displayed, with fields for 'Name' (query1), 'Data Routing URL' (http://sys1.on-state.com/OnState), 'Query Context' (CLID), and 'Default Skill' (default). There are 'Save' and 'Cancel' buttons at the bottom of the form. A note at the bottom of the screen reads: 'Note: Dialed Numbers/Names for voice calls will be phone numbers or Skype Ids. For chat calls a Dialed Number/Name can be any set of characters.'

The inputs to this screen are as follows:

The Data Routing URL is the network address to access the webservice.

The QueryContext string is passed through in the event message to indicate where the query was invoked.

The Default Skill is the skill to be invoked if not valid skill is returned within the timeout.

Note: Because call processing is delayed for the query response, the timeout is set at 4 seconds.

The delivered message is in the following format:

```
<DeliveredEvent>
  <EventType>      -- "RoutingQuery".
  <QueryContext>   -- from Query configuration—indicates where query was invoked
  <DN>             -- the Dialed Number for this call
  <CallerID>       -- the Caller ID for this call.
  <CallID>         -- the OnState unique call identifier
  <Subject>        -- the Subject field for this call.
  <Priority>        -- the Priority field (1 – 10, 10 highest)
  <Media>          -- (0 = voice, 2 = chat)
  <CreateTime>    -- time call was created in epoch seconds
  <CallType>       -- class of call
</DeliveredEvent>
```

## **RoutingQuery return values**

The response to the RoutingQuery event is expected to return a string specifying a skill configured in the OnState Supervisor. Additionally if the response to the CallOffered event is formatted with a “URL” element, then that URL value is sent to OnState Anywhere for display.

This means that the receiving system can set the information to be presented to the user for this call and return the URL to access that information in the webservice response. OnState Anywhere will then display that screen for the user.

For screenpops, the webservice response must have a Response element containing a URL element, as in the following example:

```
<Response>  
  <Skill>sales</Skill>  
  <URL>http://www.onstate.com</URL>  
</Response>
```

## CallOffered Event

This event indicates that the call has been sent to the user's phone. The userData section contains the information known about the call, including the following values. The selected user is given in the TermUser field.

```
<DeliveredEvent>
  <connection>           -- the call ID assigned to this call for tracking
  <alertingDevice>      -- the user's device. Normally the user's Skype handle or SIP address
  <callingDevice>       -- the calling line ID or other information identifying the caller
  <calledDevice>        -- the dialed number or Skype handle
  <lastRedirectionDevice> -- the last device that redirected this call
  <cause>                -- "CallOfferedEvent"
  <userData>            -- the Call Object from the switch
    <Call>
      <CallID>           -- identifier for the call
      <CreateTime>      -- time call was created (in epoch seconds)
      <Subject>          -- the subject of the call
      <Priority>         -- the call's priority
      <Media>           -- 0=voice, 2=chat
      <From>            -- original call destination (user or skill)
      <To>              -- original caller address
      <OrigUser>        -- originating user or source of call
      <TermUser>        -- terminating OnState user
      <CallType>        -- class of call
    </Call>
  </userData>
</DeliveredEvent>
```

## CallOffered Screenpop

If the response to the CallOffered Event is formatted with a URL element, then that URL value is sent to OnState Anywhere for screenpop.

As in the RoutingQuery case, this means that the receiving system can set the information to be presented to the user for this call and return the URL to access that information in the webservice response. The format for the return is

```
<Response>
  <URL>http://www.onstate.com</URL>
</Response>
```

If no URL value is found in the response, no action is taken. If URL values are provided for both the RoutingQuery and CallOffered events, then the CallOffered value will be used.

## Call Disconnect Event

This event indicates that the call has ended. Answer time and disconnect time are added to the `userData`. The answering user is again given in the `TermUser` field.

This event is provided both for calls disconnecting from users and for calls abandoned prior to answer.

```
<DeliveredEvent>
  <connection>           -- the call ID assigned to this call for tracking
  <alertingDevice>       -- the agent's device. Normally the user's Skype username or SIP address
  <callingDevice>        -- the calling line ID or other information identifying the caller
  <calledDevice>         -- the dialed number or Skype username
  <lastRedirectionDevice> -- the last device that redirected this call
  <cause>                 -- "CallDisconnectEvent"
  <userData>             -- the Call Object from the switch
    <Call>
      <CallID>           -- identifier for the call
      <CreateTime>      -- time call was created (in epoch seconds)
      <Subject>          -- the subject of the call
      <Priority>          -- the call's priority
      <Media>            -- 0=voice, 2=chat
      <SipTo>            -- original call destination (user or skill)
      <SipFrom>          -- original caller address
      <OrigUser>         -- originating user or source of call
      <TermUser>         -- terminating OnState user
      <CallType>         -- class of call
      <AnswerTime>       -- time call was answered (in epoch seconds)
      <DisconnectTime>  -- time call was ended (in epoch seconds)
    </Call>
  <userData>
</DeliveredEvent>
```

## Message Protocol

Both event messages are sent to the customer's website as Webservice calls using HTTP POST protocol with a SOAP packet in the body. The messages defined above are in the body of the SOAP packet. The soap body is UTF-8 encoded. A formal WSDL declaration is not needed.

The HTTP Protocol is documented in

[RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1](#)

The SOAP packet is documented in <http://www.w3.org/TR/soap>.

The following is an example of a message on the wire:

```
POST /callProcessing/app.dll HTTP/1.0
Host: app.yourcompany.com
Accept: */*
Accept-Language: en-us
Referer: http://skypeacd2.on-state.com/your_company/
soapaction: http://app.yourcompany.com/RouteRequest
Content-Type: application/soap_xml
User-Agent: Mozilla/4.0 (compatible)
Cache-Control: no-cache
Max-Forwards: 10
Content-Length: 478

<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:xsd=http://www.w3.org/2001/XMLSchema
  xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
  <DeliveredEvent>
    <connection>
      <alertingDevice>
      <callingDevice>
      <calledDevice>
      <lastRedirectionDevice>.
      <cause>
      <userData>
        ... and all of the other fields...
    </DeliveredEvent>
    <timestamp>1179670152328</timestamp>
  </soap:Body>
</soap:Envelope>
```

Here is what the response might look like on the wire:

```
HTTP/1.1 200 Ok
```

```
Content-Type: text/xml; charset=utf-8
```

```
Content-Length: 182
```

```
<soap:Envelope xmlns:soap=http://www.w3.org/2001/12/soap-envelope  
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">  
  <soap:Header/>  
  <soap:Body>  
    <Response>  
      <URL>http://www.on-state.com</URL>  
    </Response>  
  </soap:Body>  
</soap:Envelope>
```