VOICENTER

4

CDR Notification System

4. CDR Notification System

The CDR Notification System is a unique tool from Voicenter that lets you store all of your business telephony information, such as call detail records (CDR), and use/ access it at any time directly from your databases. Our Cloud will send you an HTTP XML-RPC requests that contain detailed information about every call.

CDR XML RPC specification

This methodology is per the XML-RPC standard, as appearing in the XML-RPC website at http://www.xml-rpc.org. Following are some examples representing an XML-RPC post and an XML-RPC response.

CDR request parameters:

#	Field	Description	Example
1	caller	Displays the caller's phone number. The number that will appear at the destination.	"caller":"0722776772"
2	target	Displays the destination of the call. Can be a phone number or the extension SIP code. The phone number value will be sent with the international country prefix.	"target":"AAPINFzL" / "target":"972722776772"
3	time	Displays the time that that the call was made in Epoch time.	"time":1536855354
4	duration	Displays the duration of the call(seconds). This duration do not include the ringing duration only the actual time of the conversation that was made.	"duration":33
5	ivruniqueid	Displays the ID code of the specific call.	"ivrunique-id":"201809131615530APIAPI-APIAac40c3d53"
6	type	Displays the type of Call. For example: if it is an incoming/outgoing call? There are several call types.	"type":"Incoming Call" / "type":"Extension Outgoing" / "type":" Click2Callleg1"
7	status	Displays what happened with the specific call? There are several call statuses.	"status":"ANSWER" / "status":"ABANDONE / "status":"TE"
8	targetextension	Displays the extension SIP code that answered to the incoming call. Sometimes this value will be identical to the "target" field. There are cases that the incoming call is received not directly to the extension, so the "target" can display different value.	"targetextension":"AAPINFzL"
9	callerextension	Displays the extension SIP code that the call was dilled from. This value is different from the "caller" field. In the "caller" field we display the actual number that will appear at the destination.	"callerextension":" AAPINFzL"
10	did	Displays the origin phone number that the caller called to.	"did":"0722776772"
11	queueid	If the call was directed to a queue service, it displays the queue code ID. In case there is no queue, the value will be 0(and not null).	"queueid":12345
12	queuename	If the call was directed to a queue service, it displays the queue name.	"queuename":"Service Queue"
13	record	Displays a URL link to the call recording.	"record":"http://starkey-centrex-recordings.s3.amazonaws.com/2018091316261401072764f-c3eb6844c-aws-AAPINF-zL-972523258000.mp3"
14	price	Displays the total price of the call in ILS cents(Agorot).	"price":7
15	dialtime	Displays the ringing duration of the call(seconds). Not include the actual conversation duration.	"dialtime":23
16	representative_name	Displays the Voicenter user name that the specific call was associated with.	"representative_name":"Walter Melon"

#	Field	Description	Example
17	representative_code	Displays the Voicenter user ID code that the specific call was associated with.	"representative_code":" 9996 "
18	targetextension_name	Displays the Voicenter extension name that answered to the specific call.	"targetextension_name":"Walter Melon"
19	callerextension_name	Displays the Voicenter extension name that this specific call was made from.	"callerextension_name":" Walter Melon"
20	target_country	Displays the country name that this outgoing call was made to.	"target_country":"Israel"
21	caller_country	Displays the country name that this incoming call was made from.	"caller_country":"Israel"
22	seconds_waiting_in_queue	This field will only be sent in the json CDR, if the specific called was directed to a queue. It displays the duration(seconds) that the caller waited in the queue.	"seconds_waiting_in_queue":5
23	OriginalIvrUniqueID	This field will only be sent in the json CDR, if the specific called was related to another call. It displays the origin call code ID. Sometimes calls are transferred in the organization.	"OriginalIvrUniqueID":" 201809131730110122APIAPIAPI-API "

Call types:

#	CDR type name	Description
1	Incoming Call	A regular outgoing call(manually dialed from the phone).
2	CC	A call that was made through a calling card (Access number) service.
3	Extension Outgoing	A regular outgoing call(manually dialed from the phone).
4	Queue	An incoming call that was received by a queue.
5	Click2Call leg1	A call that was made by click2call(Not by Dialer). Leg 1 of the call. Leg 1 - the initial connection of the call to the extension.
6	Click2Call leg2	A call that was made by click2call(Not by Dialer). Leg 2 of the call. Leg 2 - the actual call that is being made to the destination.
7	VoiceMail	A call that was answered by voicenter voicemail.
8	Callference	A call that was made through Voicenter callference service.
9	XferCDR	A call that manually transferred from an extension to a DID/another extension.
10	ProductiveCall Leg1	A "leg 1" Agents Auto Dialer calls. Leg1 - the initial connection of the call to the extension.
11	ProductiveCall Leg2	A "leg 2" Agents Auto Dialer calls. Leg 2 -the actual call that is being made to the destination.
12	Scrubber	A call that was made through Voicenter's Scrubber service. *Not released yet.
13	Click 2 IVR	"Leg1" Predictive Dialer calls. Leg1 - the initial connection of the call to the extension.
14	Click 2 IVR Incoming	"Leg 2" Predictive Dialer calls. Leg2 - the actual call that is being made to the destination.
15	Click 2 Queue Incoming	"Leg2" Predictive Dialer calls that were made through a queue. Leg2 - the actual call that is being made to the destination.
16	FaxCdr	A call that was made through Voicenter's internal outgoing fax service. *Not released yet.
17	Attended CDR leg1	A call that was transferred with consultation.
18	Attended CDR leg2	A call that was transferred with consultation. This type will only be made in a case of an incoming call that was answered by representative "A", "A" will then put the call on hold and make another call(consult) to another person - "B". Afterward, "A" will transfer the initial call to "B". The "Leg 2" is the part of the call between "B" and the initial caller.
19	Auto forward	A call that was automatically transferred from an extension to a DID(usually representatives configure their phones manually to transfer calls when they are not available).

Call statuses

#	CDR Status Name	Description
1	NOTDIALED	Hang-up occurred before the call was made.
2	ANSWER	A call is answered. A successful dial. The caller reached the callee. Whenever we receive an answer response signal, also when the call reached local voicemail service and etc.
3	BUSY	Busy signal. The dial command reached its number but the number is busy.
4	NOANSWER	No answer. The dial command reached its number, the number rang for too long, then the dial timed out
5	CANCEL	A call is canceled. The dial command reached its number but the caller hung up before the callee picked up.
6	ABANDONE	When using Voicenter's queue service, this status will appear in several cases. A caller hung up before the callee picked up. A caller while waiting in the queue, chose to exit from the queue. A call timeout in the queue.*In the future there will be a call status for each case.
7	VOEND	Hang-up during IVR without actual dialing. In this case, the caller waited in the IVR but hung up before the call rang in any extension.
8	TE	When an incoming call is directed to an IVR recording and afterward it configured to hung up the call.
9	NOTCALLED	A Leg2 Click2Call was not called. When using a click2call service and the caller hung up the call before it dialed at the destination.
10	NOTDIALED	An direct incoming call to an extension that did not reach and dialed at the extension.
11	VOICEMAIL	Call entered to Voicenter voicemail service.

	Error types	This error responses displays cases when there was a problem connecting to the target destinations. It mainly used for Voicenter internal Tracking.
12	CONGESTION	Congestion. This status is usually a sign that the dialed number is not recognized
13	CHANUNAVAIL	Channel unavailable. On SIP, peer may not be registered
14	INVALIDARGS	Error parsing dial command arguments
15	SSWPREAUTH	SSW outgoing call cancel before actual dial

972-73-279-1800 | www.voicenter.co.il | developers@voicenter.com

CDR response parameters:

#	Name	Type	Example	Description
1	err	int	0	Error code. 0 – OK 1 – Parse error 2 – Application error
2	errdesc	string	OK	Error description

EXAMPLES

Xml-rpc:

```
<?xml version="1.0"?>
<methodCall>
	<methodName>CDR</methodName>
	<params>
		<param>
			<value>
				<struct>
					<member>
						<name>caller</name>
						<value>
							<string>0540000000</string>
						</value>
					</member>
					<member>
						<name>target</name>
						<value>
							<string>050000000</string>
						</value>
					</member>
					<member>
						<name>time</name>
						<value>
							<i8>1345373417</i8>
						</value>
					</member>
					<member>
						<name>duration</name>
						<value>
							<i4>12</i4>
						</value>
					</member>
```


972-73-279-1800 | www.voicenter.co.il | developers@voicenter.com

```
                                <member>
                                        <name>ivruniqueid</name>
                                        <value>
                                                <string>sdfsdfsdfsfsf</string>
                                        </value>
                                </member>
                                <member>
                                        <name>targetextension</name>
                                        <value>
                                                <string />
                                        </value>
                                </member>
                                <member>
                                        <name>callerextension</name>
                                        <value>
                                                <string>IuYtReWq</string>
                                        </value>
                                </member>
                                <member>
                                        <name>did</name>
                                        <value>
                                                <string />
                                        </value>
                                </member>
                                <member>
                                        <name>queueid</name>
                                        <value>
                                                <i4>0</i4>
                                        </value>
                                </member>
                                <member>
                                        <name>queuename</name>
                                        <value>
                                                <string />
                                        </value>
                                </member>
                                <member>
                                        <name>type</name>
                                        <value>
                                                <string>Extension Outgoing</
string>
                                        </value>
                                </member>
                                <member>
                                        <name>status</name>
                                        <value>
                                                <string>ANSWER</string>
                                        </value>
                                </member>
                        </struct>
                </value>
        </param>
</params>
</methodCall>
```


Valid response:

```
<?xml version="1.0"?>
<methodResponse>
	<params>
		<param>
			<value>
				<struct>
					<member>
						<name>err</name>
						<value>
							<int>0</int>
						</value>
					</member>
					<member>
						<name>errdesc</name>
						<value>
							<string>OK</string>
						</value>
					</member>
				</struct>
			</value>
		</param>
	</params>
</methodResponse>
```

Error example:

```
<?xml version="1.0" ?>
<methodresponse>
	<params>
		<param>
			<value>
				<struct>
					<member>
						<name>err</name>
						<value>
							<int>2</int>
						</value>
					</member>
					<member>
						<name>errdesc</name>
						<value>
							<string>Some error description</
string>
						</value>
					</member>
				</struct>
			</value>
		</param>
	</params>
</methodresponse>
```


JSON example:

```
{
 "caller": "0544444444",
 "target": "0722776772",
 "time": 1414320455,
 "duration": 12,
 "ivruniqueid": "xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
 "type": "Extension Outgoing",
 "status": "ANSWER",
 "targetextension": "",
 "callerextension": "SIPSIPSI",
 "did": "",
 "queueid": 0,
 "queuename": "",
 "record": "http://starkey-centrex-recordings.s3.amazonaws.com/xxxxxxxxx-
aws-0544444444-0722776772.mp3",
 "dialtime": 31
}
```

Expected response:

```
{
 "err": "0",
 "errdesc": "OK"
}
```

C# web service code sample

CDR Listener implementation in C# asp.net.
We even made an example of a Simple ashx handler attached below

Simple ashx handler

Dependencies:

- CookComputing.XmlRpcV2.dll (How-to ...) (c#) (vb.net))
- Microsoft.NET Framework 4.0

Code:

```
<%@ WebHandler Language="C#" Class="CDR_Listener" %>

using System;
using System.Web;
using CookComputing.XmlRpc;

//[XmlRpcService(AutoDocumentation=false)]
public class CDR_Listener : XmlRpcService
{
	// Change it to true to get more data in output XML.
	// Change it to false in productive system.
	private bool isDebug = true;

	private void HandleRequest(CdrData cdr)
	{
		/* ADD YOUR CODE HERE */

	}
```



```
19
20          [XmlRpcMethod()]
21          public XmlRpcStruct CDR(XmlRpcStruct data)
22          {
23                  XmlRpcStruct ret = new XmlRpcStruct();
24                  if(this.isDebug)
25                          ret.Add("origrequest", data);
26                  try
27                  {
28                          CdrData cdr = new CdrData()
29                          {
30                                  Callerid = GetValue<string>("caller", string.Empty, data),
31                                  Target = GetValue<string>("target", string.Empty, data),
32                                  Duration = GetValue<int>("duration", 0, data),
33                                  IvrUniqueID = GetValue<string>("ivruniqueid", string.Empty,
34  data),
35                                  Epoch = GetValue<long>("time", 0, data),
36                                  extensionCaller = GetValue<string>("callerextension", string.
37  Empty, data),
38                                  extensionTarget = GetValue<string>("targetextension", string.
39  Empty, data),
40                                  status = GetValue<string>("status", string.Empty, data),
41                                  did = GetValue<string>("did", string.Empty, data),
42                                  type = GetValue<string>("type", string.Empty, data),
43                                  queueID = GetValue<int>("queueid", 0, data),
44                                  queueName = GetValue<string>("queuename", string.Empty, data)
45                          };
46                          if (this.isDebug)
47                                  ret.Add("got", cdr);
48
49                          try
50                          {
51                                  HandleRequest(cdr);
52                          }
53                          catch (Exception ex)
54                          {
55                                  if (this.isDebug)
56                                          throw new CustomerException(ex, "Customer code
57  exception");
58                                  else
59                                          throw new CustomerException(new Exception("Oops...
60  something wrong"), "Customer code exception");
61                          }
62                  }
63                  catch (CustomerException ex)
64                  {
65                          ret.Add("err", "2");
66                          ret.Add("errdesc", ex.OriginalException.ToString() + " >>> " +
67  ex.Description);
68                          return ret;
69                  }
70                  catch (Exception ex)
71                  {
72                          ret.Add("err", "1");
73                          ret.Add("errdesc", ex.ToString());
74                          return ret;
75                  }
76
77                  ret.Add("err", "0");
78                  ret.Add("errdesc", "OK");
79
80                  return ret;
```



```
81          }
82
83
84          private T GetValue<T>(string key, T defaultValue, XmlRpcStruct data)
85          {
86                  key = key.Trim();
87                  if (!data.ContainsKey(key))
88                          return defaultValue;
89                  T value;
90                  try
91                  {
92                          value = (T)Convert.ChangeType(data[key], typeof(T));
93                  }
94                  catch
95                  {
96                          value = defaultValue;
97                  }
98                  return value;
99          }
100 }
101
102
103
104
105
106 public class CdrData
107 {
108         public string Target = string.Empty;
109         public string Callerid = string.Empty;
110         public string IvrUniqueID = string.Empty;
111         public long Epoch = 0;
112         public int Duration = 0;
113         public DateTime Time { get { return new DateTime(1970, 1, 1, 0, 0, 0,
114 DateTimeKind.Unspecified).AddSeconds(this.Epoch); } }
115          public string extensionTarget;
116         public string extensionCaller;
117         public string did;
118         public string callType;
119         public string callStatus;
120         public int queueID;
121         public string queueName;
122
123 }
124
125
126 public class CustomerException : Exception
127 {
128         public Exception OriginalException { get; private set; }
129         public string Description { get; private set; }
130
131         public CustomerException(Exception OriginalException, string Description)
132         {
133                 this.OriginalException = OriginalException;
134                 this.Description = Description.Trim();
135         }
136 }
```

To receive additional value: (REQUIRES IMPLEMENTATION!!!)

1. Add field to CdrData class.
2. "CdrData cdr = new CdrData() { ... // Add parsing row HERE }".
3. Check that the XML includes the new value to be able to get it.

