New Rock Technologies, Inc.

OM API

Technical Development Document

Website: http://www.newrocktech.com/

Tel: +86 21-61202700

Fax: +86 21-61202704

Document Version: 5.0

Release Time: May 2017

OM API Technical Development Document（V5.0）

This article mainly introduces the function and method of OM API, which helps developers to learn and master it, and eventually develop more and better applications.

Something you need to know:

Before reading this document, you'd better know something about our OM series of IPPBX products.

Recommended materials: 《OM user manual》, 《OM administrator manual》, 《OM function learning guide》.

Materials download: OM related materials can be downloaded from the official website.。

Explanation: the OM API is the same on different models of products, but the version is different.

1 Quick Start

The chapter introduces basic concepts and interaction principle between Server and OM, which directs you to complete the OM API authentication configuration. And then show you how to use the test tool to obtain interaction messages between the application server and the OM, to help you get started quickly.

1.1 First Meeting

About OM API

The IPPBX devices of OM series offer a unified open interface - OM API, which allows developers to control, monitor, and statistical analysis via the OM API.

OM API is essentially an encapsulated simple XML message that communicates between the application server and the OM device through the HTTP protocol.

1. **The application server operates the OM through the OM API：** Parameter Query, Parameter Configuration, Status Query, Call Control, Call Transfer, etc.

2. **OM pushes reports to the application server in real time：** Extension Status Changes, Call Status Changes, Configuration Changes, Service Startup, DTMF, Voice Play Finished, Call Logs, etc.

OM API makes the OM device more flexible and operational, and developers can use OM API to develop or connect the Call Center, Billing Systems, PMS, CRM, OA Systems and other colorful applications.

There are four skills：

Parameter and status query

Parameter configuration

Call control

Event monitoring

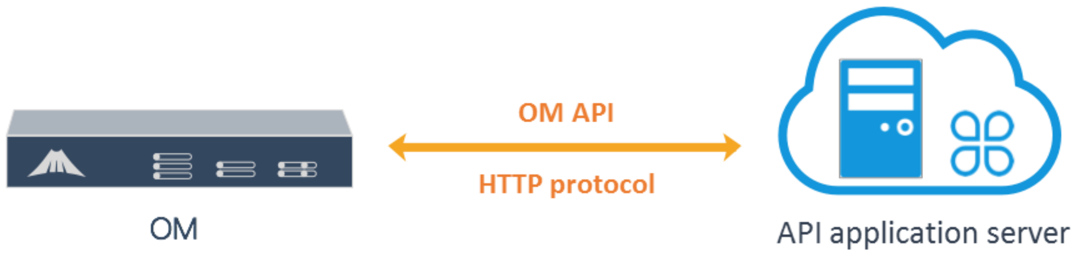
The main functions： Click to dial, screen popup, call recording, recording, monitoring, group and queue, IVR voice navigation, satisfaction survey, hotel wake-up call service, voice mail, voice verification code, call the black/white list.

1.2 Interaction principle

After a brief knowledge of OM API, let's get to know how OM API communicate.

Transport protocol

Communication between the application server and the OM is based on the HTTP protocol, and the API message is encapsulated in the HTTP package.



Message contents such as:

HTTP request message：

```
POST /xml HTTP/1.0
Content-Type:text/xml
Content-Length:101

<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
      <DeviceInfo/>
</Control>
```

HTTP response message：

```
HTTP/1.0 200 OK

<?xml version="1.0" encoding="utf-8" ?>
  <DeviceInfo>
      <manufacturer>New Rock Technologies, Inc</manufacturer>
      <model>Rev 1.0.1 WROC2000-1S/1</model>
```

```
    <version>Rev 2.2.5.81.1</version>
    <mac>00:0E:A9:00:12:BD </mac>
    <devices>
        <ext lineid="Phone 1" id="200" />
        <ext lineid="IPPhone 50" id="208" />
        <line lineid="Line 2" id="02161208234" />
        <line lineid="IPLine 21" id="02161204000" />
    </devices>
</DeviceInfo>
```

Communication mode

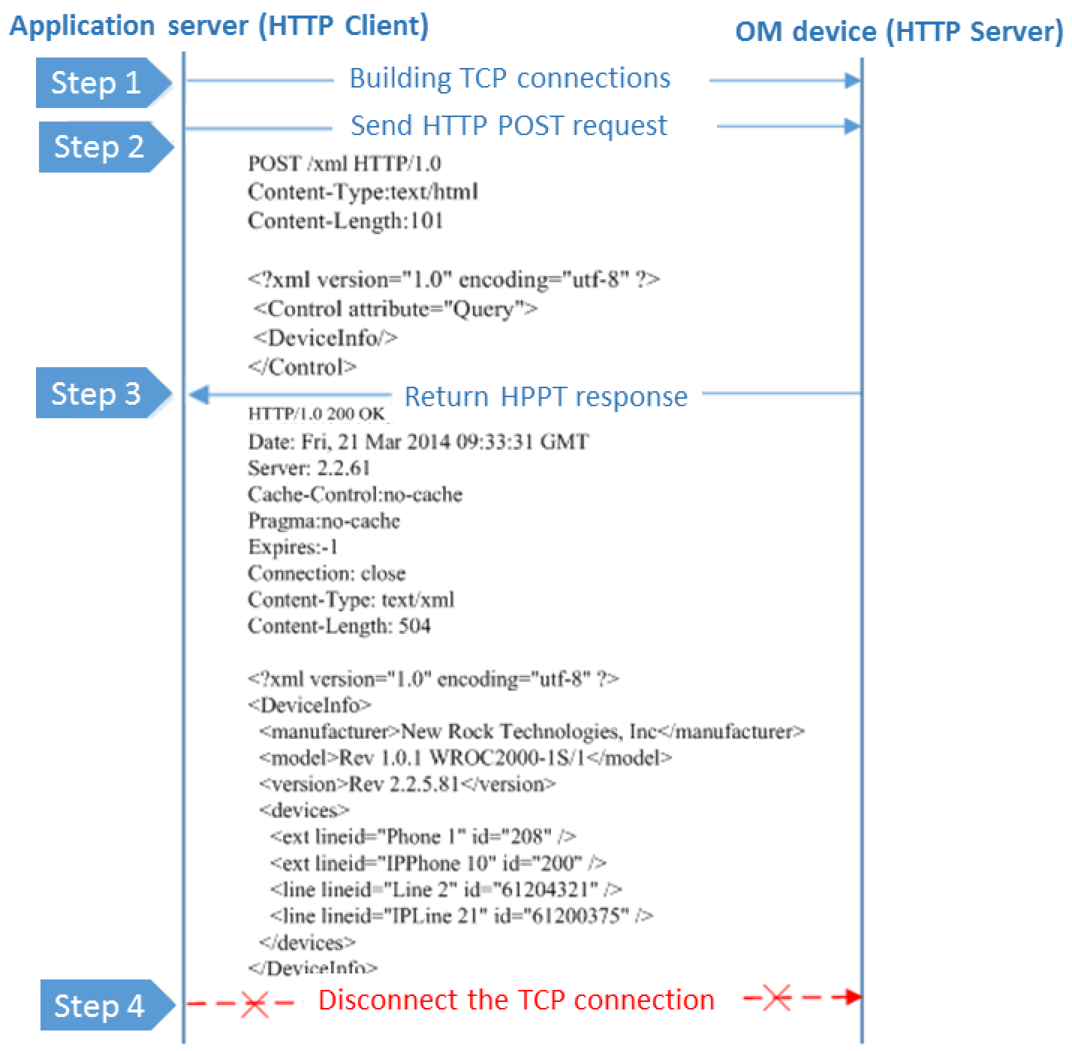
The interaction between application server and OM is bidirectional, and the two sides act as HTTP server and client each other.

The one case：The application server acts as the HTTP client, and the OM serves as the HTTP server.

The application server requests the OM to perform a function (such as initiating a call) or provide certain information (e.g., query status). At this point, the HTTP POST method is adopted.

The procedure is: request, response, disconnection.

The interaction diagram follows：



```
POST /xml HTTP/1.0
Content-Type:text/html
Content-Length:101

<?xml version="1.0" encoding="utf-8" ?>
 <Control attribute="Query">
 <DeviceInfo/>
</Control>
```

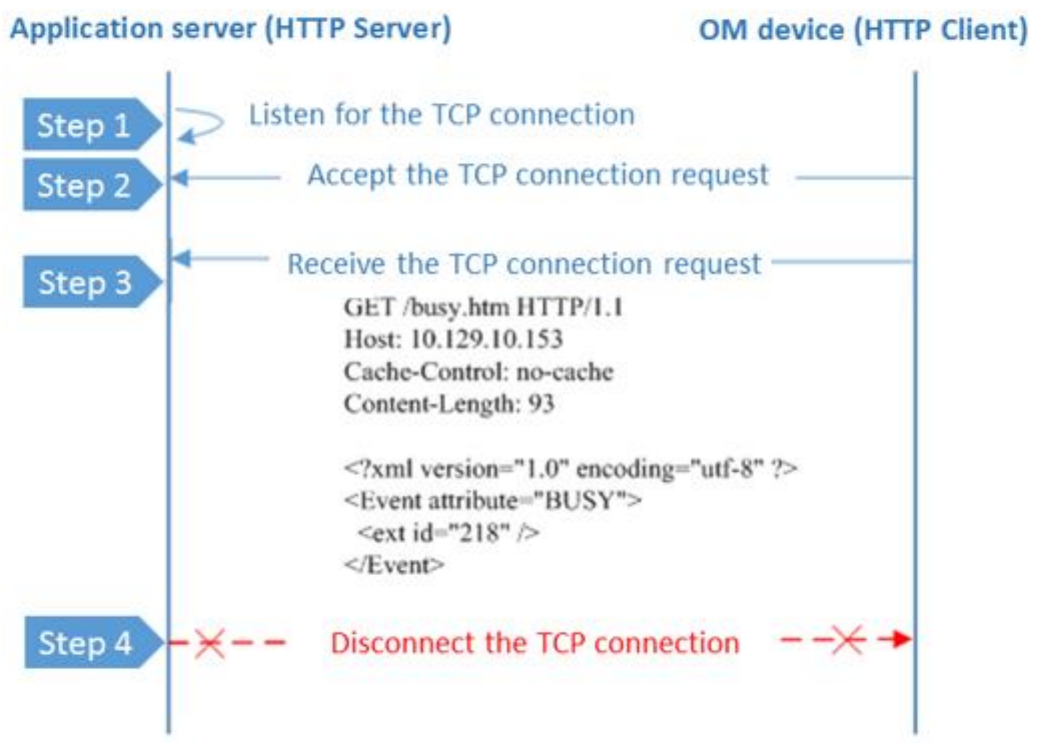
```
HTTP/1.0 200 OK
Date: Fri, 21 Mar 2014 09:33:31 GMT
Server: 2.2.61
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Connection: close
Content-Type: text/xml
Content-Length: 504

<?xml version="1.0" encoding="utf-8" ?>
<DeviceInfo>
  <manufacturer>New Rock Technologies, Inc</manufacturer>
  <model>Rev 1.0.1 WROC2000-1S/1</model>
  <version>Rev 2.2.5.81</version>
  <devices>
    <ext lineid="Phone 1" id="208" />
    <ext lineid="IPPhone 10" id="200" />
    <line lineid="Line 2" id="61204321" />
    <line lineid="IPLine 21" id="61200375" />
  </devices>
</DeviceInfo>
```

The other case：OM acts as the HTTP client, the application server acts as the HTTP server

OM pushes some messages (e.g., ring events), the application server receives the messages and disconnect TCP (Note: This is not a standard HTTP request and response process, so OM does not need the application server reply response).

At this point, GET or POST method is used (default is GET, and if the parameter API_METHOD = 1, which is POST).

The procedure is: receive message and disconnect.


Application server (HTTP Server)
OM device (HTTP Client)
Step 1
Listen for the TCP connection
Step 2
Accept the TCP connection request
Step 3
Receive the TCP connection request
Step 4
Disconnect the TCP connection

```
GET /busy.htm HTTP/1.1
Host: 10.129.10.153
Cache-Control: no-cache
Content-Length: 93

<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BUSY">
  <ext id="218" />
</Event>
```

1.3 Demo

Next, we demonstrate the configuration methods of OM and API application servers and the process of sending and receiving messages using a test tool (equivalent to a simple application server).

Configure

Note: We use IP authentication here.

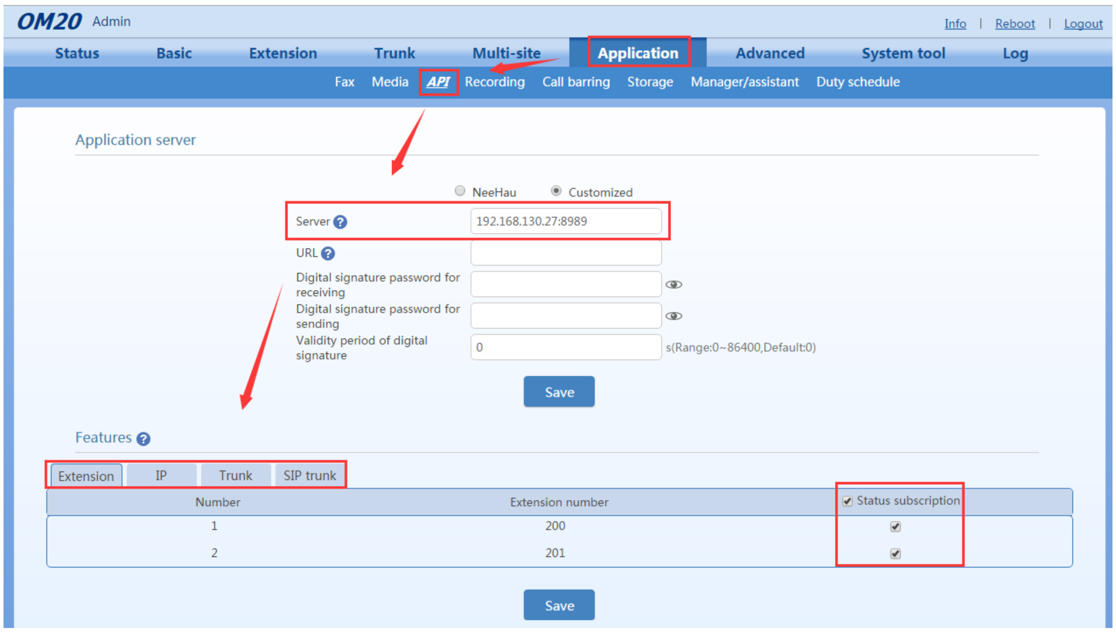
Step 1: Configure the OM device authentication address

Login OM, click on the **application > API**, select the customized column in the application server (selected by default for NeeHau, is built in the device as a small call center), fill in the application server address (i.e. IP:Port, where the tool can be customized) such as 192.168.130.27:8989.

Step 2: Configure the API switches for extensions and trunks

Enter the **application > API**, in the column of features, open the switchs of pre-answer control for incoming, post-answer control for incoming, status subscription on the extension and trunk, click save, and restart the OM device.

The OM device configuration is follow:



Application server

( ) NeeHau [✓ Customized]

Server	192.168.130.27:8989
URL	
Digital signature password for receiving	
Digital signature password for sending	
Validity period of digital signature	0 s(Range:0~86400,Default:0)

Save

Features

Extension | IP | Trunk | SIP trunk

Number	Extension number	✓ Status subscription
1	200	✓
2	201	✓

Save

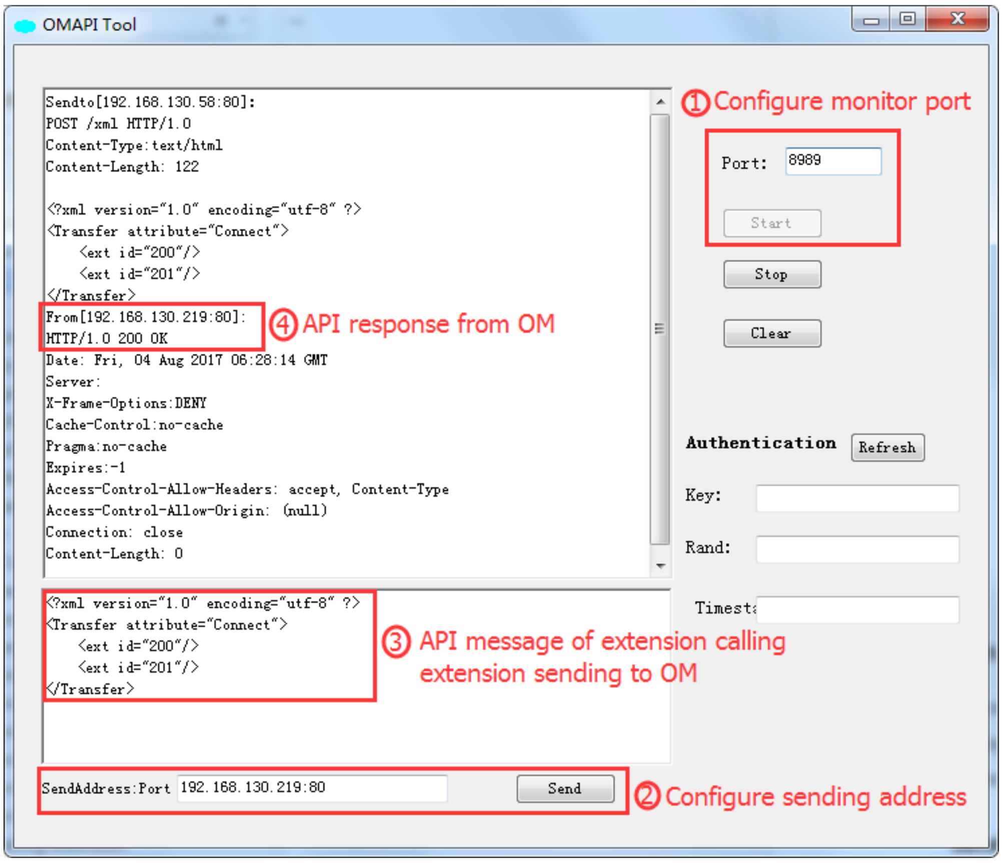
Step 3: Configure the test tool

1. Download the test tool and open it on your computer desktop to fill in a HTTP listening port (the port is identical to the OM sending port configured on the OM device, may be 8989) and click start listening.

2. Fill in the sending address (i.e., the IP address and port of OM), such as 192.168.130.219:80.

Note： The HTTP port of OM defaults 80.

The test tool configuration is follow：



```
Sendto[192.168.130.58:80]:
POST /xml HTTP/1.0
Content-Type:text/html
Content-Length: 122

<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="200"/>
    <ext id="201"/>
</Transfer>
From[192.168.130.219:80]:
HTTP/1.0 200 OK
Date: Fri, 04 Aug 2017 06:28:14 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
```

Port: 8989

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="200"/>
    <ext id="201"/>
</Transfer>
```

SendAddress:Port 192.168.130.219:80

Run a process

Since the configuration has been completed, we demonstrate the process of an extension calling an extension to lays the foundation for the use and understanding of the interface in the Chapter 2.

Send command

How to implement extension calling extension through OM API?

Just send an API message to OM：

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
  <ext id="200"/>
  <ext id="201"/>
</Transfer>
```

Explanation：

- The first row is the XML statement that defines the XML version (1.0) and the character encoding used (UTF-8). It is the same for every API message.

- The second line is the root of the XML. The root element “Transfer “ indicates that this is a call forwarding class of API. The attribute value “Connect” indicates the property of this transit as a connection.
- In the third and fourth line, the “ext” is the abbreviation of extension. The 200 is the calling number, and the 201 is called number.
- The fifth line is the closing tag of root node.

Note: For more OM API syntax, please see the second chapter.

Observe the execution results

After the execution of the API, the following process is：

1. The calling extension 200 will ring first (by default, who calls first, who calls later, which is controlled by parameter API_CALLING).
2. After the extension 200 offhooked, the called extension 201 rings, and the 200 can hear a ringback tone of "beep".
3. After the 201 offhooked, both sides establish a call.
4. After any party hangs up, the call is over.

Observe incoming API messages

Looking at the test tool, you can see that a lot of API messages are received. The messages consist of two types, namely, CDR and event.

- CDR: call records, produced at the end of the conversation. One of the CDRs is the call record of the calling extension, and the other is that of the called extension.
- Event: event message, triggered by OM automatically in call process.

1）Event

Attributes of the events triggered during this call include：BUSY、IDLE、RING、ALERT、ANSWER、ANSWERED、BYE.

Among them：

- The BUSY and IDLE are a pair, produced when the extension state changes. BUSY indicates that the extension is changed from idle to busy, and IDLE indicates that the extension is changed from busy to idle.

- RING、ALERT、ANSWER、ANSWERED and BYE belong to one category, which are triggered when the call state changes..

  Among them：

  - RING and ALERT are a pair. RING indicates that the extension starts ringing, and the ALERT indicates the call back (ringback) signal is received.
  - ANSWER and ANWERED are a pair. ANSWER indicates the extension answer, and ANSWERED indicates the answer signal of the other side is received.
  - BYE indicates the end of call.

Through these events, you can monitor the state of extension line and the call in real time. Also you can achieve some application functions, such as screen popup of incoming and outbound call (when the extension ring, the software screen popup, and corresponding customer informations are displayed).

Note: For more information about events, please refer to Chapter 2.4.

2）Call record（CDR）

At the end of the call, OM immediately pushes the call record to the test tool (application server).

Message format

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="13620170308103713-0">
   <callid>32820</callid>
   <TimeStart>20170308103709</TimeStart>
   <Type>IN</Type>
   <Route>IC</Route>
   <CPN>200</CPN>
   <CDPN>201</CDPN>
   <TimeEnd>20170308103713</TimeEnd>
   <Duration>2</Duration>
   <TrunkNumber></TrunkNumber>
   <Recording>20170308/200_201_20170308_103711_8034_cd.wav</Recording>
   <RecCodec>PCMU</RecCodec>
</Cdr>
```

Parameter description

Parameter name	**Interpretative statement**
Cdr id	Number of call record. Format: number + date + fixed content (-0 or -1 etc)
callid	Relative unique identifier for call
TimeStart	The starting timestamp of a call, format: YYYYMMDDHHMISS
Type	Traffic type, IN indicates inbound call, and LO indicates internal call
Route	The routing type, IC represents the internal routing
CPN	Calling number
CDPN	Called number
TimeEnd	Release timestamp of a call, format: YYYYMMDDHHMISS
Duration	Length of a call, unit: sec. That is, the duration of the call, excluding the ringing time.
Trunk	The trunk number (This is an internal call, so the value is empty)
Recording	The relative conservation path of a recording file, format: YYYYMMDD/record file name

Note: For more information about CDR, please refer to Chapter 2.5.

2 Interface

This chapter contains two parts of the OM API interface, including the API request command and the API report.

- API request command: Refers to the API messages sent by the application server to the OM, including three types, control command, transfer command, and call acceptance command.
- API report: Refers to the API message pushed by the OM to the application server, including two types, event reporting and call record reporting.

2.1 Control command

Control commands include query, config, hold, unhold, mute, unmute, monitor, bargein and clear.

2.1.1 Query

This type of API is used to query some relevant information of objects specified on the OM device (such as configuration parameters and status). These objects include device information (deviceInfo), extension (ext), trunk line (trunk), incoming call (visitor), outbound call (outer), extension group (group), and voice menu (menu).

Rule description for query request

- The smallest unit is a query object, which does not support separate query a specific parameter of the object

Rule description of query results

- There are all the relevant parameters and the status information of the object in the query results.
- If the query results do not carry a parameter information, it may be:
    - This parameter has a default value.
    - This parameter does not exist.
    - This parameter cannot be queried.

Query device information

The API is used to query the relevant information of the OM device itself, such as: manufacturer, hardware version, software version, and all extensions, trunks and extension group.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
    <DeviceInfo/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<DeviceInfo>	string	Basic information about the OM device	No value

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<DeviceInfo>
    <manufacturer>New Rock Technologies, Inc</manufacturer>
    <model>Rev 6.0.0 OM20-2S/2</model>
    <version>Rev 2.1.5.111</version>
    <mac>00:0E:A9:00:12:BD </mac>
    <devices>
        <ext lineid="Phone 1" id="200" />
        <ext lineid="IPPhone 50" id="208" />
        <line lineid="Line 2" id="02161208234" />
        <line lineid="IPLine 21" id="02161204000" />
        <group id="1">
            <ext id="200" />
        </group>
    </devices>
</DeviceInfo>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	**Type**	**Parameter description**	**Parameter value specification**
DeviceInfo	object	OM device information	Nothing
<manufacturer>	string	Manufacturer	New Rock Technologies, Inc
<model>	string	Hardware version	as Rev 6.0.0 OM20-2S/2
<version>	string	Software version	as Rev 2.1.5.111
[mac]	string	Physical address	as 00:0E:A9:00:12:BD(Determines whether the MAC address is carried by the system parameter API_MAC)
[ext]	object	Extension	
<ext lineid>	string	Extension's line number, the only fixed identifier of an extension.	<Phone | IPPhone> {NO.}, as：Phone 1
<ext id>	string	Extension number	
[line]	object	Trunk	Line and trunk refers to the same object
<line lineid>	string	Trunk's line number, the only fixed identifier of a trunk.	<Line | IPLine> { NO.},as：Line 13
[line id]	string	Trunk number	
[group id]	int	Extension group number	1~50

Note:

- The response result contains all the extension and trunk information, if a large amount lines of equipment, pay attention to the query cache space received.
- The equipment has a large amount lines, it is recommended to use web interface to obtain the line information of extension and trunk.

Query extension

The API is used to query the relevant information of an extension, such as configuration parameters, extension status, caller, etc.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
    <ext id="208"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<ext id>	string	Extension number	Must be valid extension on OM. Value cannot be null.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <ext id="208">
        <lineid>Phone 1</lineid>
        <group id="1"/>
        <group id="2"/>
        <staffid>1304081</staffid>
        <Call_Pickup>yes</Call_Pickup>
        <Fwd_Number>18603752801</Fwd_Number>
        <Call_Restriction>3</Call_Restriction>
        <Off_Line_Num>200</Off_Line_Num>
        <mobile>18603752800</mobile>
        <fork>18603752802</fork>
```

```
            <email>admin@hotmail.com</email>
            <record>on</record>
            <api>7</api>
            <voicefile>welcome</voicefile>
            <state>active</state>
            <outer id="8" from="208" to="13012345678" trunk="02161208234"
callid="28680">
                  <state>talk</state>
            </outer>
      </ext>
</Status>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Parameter description	Parameter value specification
<ext id>	string	Extension number	Pure digital string
<lineid>	string	Extension line number , the only fixed identifier of an extension.	IPPhone | Phone XXX
[staffid]	string	Job number, the operator will always play the staff number before the extension is connected	Pure digital string
[group]	int	extension group number, the extension belongs to	1~50
[voicefile]	string	Voice file, played while waiting in the extension queue	Only.Dat and.Pcm formats are supported.

Parameter name	Type	Parameter description	Parameter value specification
[email]	string	Employee email address	Not support
[Call_Restriction]	int	Call authority	0: inside line 1: local call 2: domestic 3: International
[Call_Pickup]	string	Access rights. Whether calls can be allowed to be answered by other extensions.	Yes:allow，no:refuse
[No_Disturb]	string	No disturb function switch. After opening no disturb, the extension will block all incoming calls, but can initiate the call automatically.	on: open，off: close
[Fwd_Type]	int	Call forward mode	0: close， 1: all forward， 2: forward when busy or no answer
[Fwd_Number]	string	Call forward number	Close when value is empty
[fork]	string	Group vibration number	Close when value is empty
[mobile]	string	The mobile phone number of extension, can be used as a default configuration of call forwarding or offline forwarding.	Close when value is empty
[record]	string	Recording function	on: open，off: close

Parameter name	Type	Parameter description	Parameter value specification
		switch in real time	
[api]	int	API function switch	0:close API status monitoring 7: open API status monitoring
[state]	string	Line status	Ready: Free available Active: Ringing, ringing back, or talking Progress: Analog extension offhook, dialing and process after dialling Offline: IP extension offline Offhook: The state of an analog extension when listening to a howler tone
[outer]	object	Outbound call	
[id]	int	Outbound call number, can be used to transfer, query or hang up the call, etc.	
[from]	string	Original caller’s number	
<to>	string	Original callee’s number（For outer, the primitive called number is the trunk number of the outbound call）	
[trunk]	string	Trunk number. The outbond call will be out	

Parameter name	**Type**	**Parameter description**	**Parameter value specification**
		from the trunk.	
[callid]	int	Relative unique identifier for call	
[state]	string	State of call	Talk: Call in progress Progress: Call processing Wait: Call waiting
[visitor]	object	Incoming call, as the caller of this inquiry extension	
<id>	int	Incoming call number, can be used to transfer, query or hang up the call, etc.	
<from>	string	Original caller’s number	
<to>	string	Original callee’s number （For visitor, the primitive called number is the trunk number of the incoming call）	
<callid>	int	Relative unique identifier for call	
[state]	string	State of call	Talk: Call in progress Progress: Call processing Wait: Call waiting
[ext]	object	Extension,as the caller or callee of this inquiry extension	

Parameter name	Type	Parameter description	Parameter value specification
<id>	string	Extension number	
[state]	string	State of call	Talk: Call in progress Progress: Call processing Wait: Call waiting

Query trunk

The API is used to query the pertinent information of a trunk, such as configuration parameters, line status, call status, etc.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
      <trunk id="2174"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<trunk id>	string	Trunk number	Must be valid on the OM. The value cannot be null.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
      <trunk id="2174">
            <lineid>Line 75</lineid>
            <state>active</state>
            <visitor id="2" from="202" to="2174" callid="36866">
                  <state>talk</state>
            </visitor>
      </trunk>
</Status>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	**Type**	**Parameter description**	**Parameter value specification**
<trunk id>	string	Trunk number	Digital string
<lineid>	string	Trunk's line number , the only fixed identifier of a trunk.	IPLine | Line XXX
<state>	string	Trunk line status	ready: available active: offhook,ringingor talking unwired: no wiring offline: Off-line Note: For IP trunk multiplexing, the state is ready as long as there is one free space available.
[outer]	object	Outer call	
<id>	int	Outbound call number, can be used to transfer, query or hang up the call, etc.	
<from>	string	Original caller's number	
<to>	string	Original callee's number（For outer, the primitive called number is the trunk number of the outbound call）	
<trunk>	string	Trunk number. The	

Parameter name	**Type**	**Parameter description**	**Parameter value specification**
		outbound call will be out from the trunk.	
<callid>	int	Relative unique identifier for call	
<state>	string	State of call	Talk: Call in progress Progress: Call processing Wait: Call waiting
[visitor]	object	incoming call,as the caller of this inquiry extension	
<id>	int	Incoming call number, can be used to transfer, query or hang up the call, etc.	
<from>	string	Original caller’s number	
<to>	string	Original callee’s number（For visitor, the primitive called number is the trunk number of the incoming call）	
<callid>	int	Relative unique identifier for call	
<state>	string	State of call	Talk: Call in progress Progress: Call processing Wait: Call waiting

Query incoming call

The API is used to query the relevant information of an incoming call, such as caller's attribute parameters (number, initial caller, original callee, call status, relative unique identifier), caller's calling party, and call status.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
    <visitor id="1"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<visitor id>	int	Visitor number	Digital. When the value is empty, all incoming calls will be displayed.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <visitor id="1" from="02167103750" to="02161208234" callid="49189">
        <ext id=" 200" />
        <state> progress </state>
    </visitor>
</Status>
```

Explanation: the incoming caller 1 is inbound from the external telephone 02167103750 via the trunk 02161208234 to the OM device and is currently in communication with extension 200.

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Description

Parameter name	Type	Description
[visitor]	object	Incoming call, an external who calls from trunk
<id>	int	Incoming call number, can be used to transfer, query or hang up the call, etc.
<from>	string	Original calling number
<to>	string	The original callee's number (For visitor, the primitive called number is the trunk number of the incoming call)
<callid>	int	Relative unique identifier of a call
[ext | menu | outer]	object	The caller may be the extension, voice menu, call storage area, broadcast area, outer call. If it's empty, indicating that the incoming call has not been forwarded to OM.
[state]	string	Talk: Call in progress Progress: Call processing Wait: Call waiting

Query outbound call

The API is used to query the relevant information of an outbound call, such as the call's number, the caller, the callee, the trunk number, the call status, and the relative unique identifier of the call.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
    <outer id="5"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<outer id>	int	The number	Digital, when the value is empty, list all

Parameter name	Type	Parameter description	Parameter value specification
		of the outbound call	outer call

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <outer id="5" from="200" to="13012345678" trunk="02161208234" callid="32773">
        <outer id="5" from="200" to="13012345678" trunk="02161208234"
callid="32773"/>
    </outer>
</Status>
```

Explanation: the outer call 5 is a call initiated by the extension 200 and external telephone 13012345678, and the trunk number is 02161208234.

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Description
[outer]	object	Outer call, from the trunk of OM device
<id>	int	Outbound call number, can be used to transfer, query or hang up the call, etc.
<from>	string	Original calling number
<to>	string	The original callee's number (For outer, the primitive called number is the trunk number of the outbound call)
<trunk>	string	Trunk number
<callid>	int	Relative unique identifier for call
[state]	string	Talk: Call in progress Progress: Call processing

Parameter name	Type	Description
		Wait: Call waiting

Query group

The API is used to query the relevant information of an extension group, such as configuration parameters (the member of the extension, the background music played during call queuing, the call distribution rules), and the incoming calls waiting in the queue of the crew.

Note: The queue order in the extension group queue is from top to bottom.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
       <group id="1"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<group id>	int	Extension group's number	1~50, When the value is empty, all the extension groups will be displayed.

Response example

```
<?xml version="1.0" encoding="UTF-8"?>
<Status>
       <group id="1">
              <ext id="200"/>
              <ext id="208"/>
              <voicefile>NowMorning</voicefile>
              <distribution>sequential</distribution>
              <visitor id="27" from="02167103750" to="02161208234" callid="49162"/>
              <visitor id="28" from="13012345678" to="02161204000" callid="49164"/>
       </group>
```

```
</Status>
```

Explanation: There are two extension members (extension 200 and extension 208) in extension group 1. When the call is transfered to the extension group, the branch will assign the call to extension according to the type of distribution (call distribution rule). When all extensions in the group is busy, the incoming call will wait in the queue and the waiting music is the voice file specified in voicefile field.

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	**Type**	**Description**
<group id>	int	Extension group’s number
[voicefile]	int	Voice files, support two formats of DAT and PCM, for the call waiting to play
[distribution]	string	Valid values: sequential, circular, group vibration, default value: circular
[ext]	string	Extension, this is the extension member of the extension group
[visitor]	object	Incoming call, waiting in the queue of an extension group
<id>	int	Incoming call number, can be used to transfer, query or hang up the call, etc.
<from>	string	Original calling number
<to>	string	The original callee number (For visitor, the primitive called number is the trunk number of the incoming call)
<callid>	string	Relative unique identifier of the call.

Query voice menu

The API is used to query the relevant information of a voice menu, such as configuration parameters (voice files, dial length, key check terminator), call messages to the menu, etc.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Query">
    <menu id="1"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description	Parameter value specification
<menu id>	int	Number of voice menu	1~50. When the value is empty, all the speech menus will be displayed.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <menu id="1">
        <voicefile>welcome</voicefile>
        <repeat>3</repeat>
        <infolength>5</infolength>
        <exit>#</exit>
        <outer id="44" from="200" to="02167103750" trunk="02161208234"/>
        <visitor id="46" from="13012345678" to="02161204000"/>
    </menu>
</Status>
```

Explanation: The caller 13012345678 and the callee 02167103750 are transferred to the voice menu 1, which plays a voice file named welcome. In addition, if the call dial key length reached 5, or enter the # number, then OM will push the key information of DTMF event report to the application server.

Note: When no call was diverted to the menu, there will be XML annotation fields: <! -- Empty Waiting List -- > in the response body, which is a normal phenomenon.

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	**Type**	**Description**
<menu id>	int	The menu number, used for querying, configuring, transfering and other operations to judge the basis.
[voicefile]	string	Voice file, the OM broadcasts the file to the caller when the call is transfered to the menu and connected. (only DAT and PCM formats are supported)
[repeat]	int	The number of times the voice file is played, the range is 0~50, and the value 0 is for loop play.
[infolength]	int	Dial detection length. When the key length reaches this length, OM pushes the counted key information (DTMF event) to the application server and starts the statistics again.
[exit]	char	Keystroke check Terminator. When the caller of the menu dials the character, OM immediately pushes the key information (DTMF event) to the application server and starts the statistics again.
[visitor]	object	Incoming call (incoming call from a trunk), transferred to the voice menu.
<id>	int	Incoming call number, can be used to transfer, query or hang up the call, etc.
<from>	string	Original calling number
<to>	string	The original callee number (For visitor, the primitive called number is the trunk number of the incoming call)
<callid>	int	The relative unique identifier of the call (**Note:** Its value can be recycled and can not be used as a permanent identifier)
[outer]	object	Outer call, the call transferred to the voice menu.
<id>	int	Outbound call number, can be used to transfer, query or hang up the call, etc.
<from>	string	Original calling number

Parameter name	Type	Description
<to>	string	The original callee number (For outer, the primitive called number is the trunk number of the outbound call)
<trunk>	string	Trunk number
<callid>	int	The relative unique identifier of the call (**Note:** Its value will be recycled and cannot be used as a permanent identifier)

2.1.2 Assign

This type of API is used to configure parameters for OM's objects, which include voice menu (menu), extension (EXT), extension group (Group) and so on.

Rule description of configuration request

- **The configuration instructions of the extension：** Eeach configuration parameters of extension are independent, that is, if there is no some parameter in configuration comman, the parameter value keeps the original.
- **The configuration instructions of other objects：** In addition to the extension, the configuration parameters of other objects are not independent, that is, if the configuration command does not carry some parameter, the parameter value may be emptied or restored the default value.

Rule description of configuration results

- If the parameter value is the default or null, there may be no the parameters in the response results.
- If the configuration parameters are not effective, there may be no the parameters in the response results.
- In addition, the results will contain all possible configuration parameters and state information, etc.

Assign extension

The API is used to configure parameters of an extension specified line number, and return the results and status configured.

Configuration parameters include extension number, call status mode, call authority, no disturb, and access rights, etc.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Assign">
    <ext lineid="IPPhone 50">
        <id>208</id>
        <staffid>1304081</staffid>
        <mobile>18603752800</mobile>
        <group>1</group>
        <group>2</group>
        <voicefile>busy</voicefile>
        <email>admin@hotmail.com</email>
        <Call_Restriction>3</Call_Restriction>
        <Call_Pickup>no</Call_Pickup>
        <No_Disturb>no</No_Disturb>
        <Fwd_Type>0</Fwd_Type>
        <Fwd_Number>18603752801</Fwd_Number>
        <Fork>18603752802</Fork>
        <WAKEUP>06:30</WAKEUP>
        <record>on</record>
        <autoAnswer>yes</autoAnswer>
        <api>7</api>
    </ext>
</Control>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Parameter description	Parameter value specification
<ext lineid>	string	Extension line number, can be obtained by querying device	<Phone | IPPhone> {number}, as Phone 1

Parameter name	**Type**	**Parameter description**	**Parameter value specification**
		information.	
[id]	string	Extension number	Pure digital non empty strings
[staffid]	string	Job number. The operator will always play the number before the extension is connected.	A pure numeric string that does not take effect when the value is empty.
[group]	int	The number of the extension group the extension belongs to. The same extension may belong to several groups.	1~50
[voicefile]	string	Voice file, played while waiting in the extension queue.	Only Dat and Pcm formats are supported.
[email]	string	Employee email address	Not support
[Call_Restriction]	int	Call authority	0: internal, 1: local, 2: domestic, 3: International
[Call_Pickup]	string	Access rights, decide whether or not the call can be answered by other extensions.	Yes: allow，no: refuse
[No_Disturb]	string	No disturb function switch. After being opened no disturb function, the extension will block all incoming calls, but can initiate the call actively.	on: open，off: close

Parameter name	Type	Parameter description	Parameter value specification
[Fwd_Type]	int	Call forward mode	0: close，1: all forward，2: forward when busy or no answer
[Fwd_Number]	string	Call forward number	Close when value is empty
[Fork]	string	Call forking number	Close when value is empty
[mobile]	string	Mobile phone number. Outbound call will be allowed if the calling party number matches the number set here or the device will notify the caller for PIN verification.	Close when value is empty
[WAKEUP]	string	Wake up time of alarm clock, 24 hour system, accurate to minutes.	The format is HH:MM and the alarm is switched off when the value is 00:00
[record]	string	Real-time recording switch	on: open, off: close, the default value is off.
[api]	int	API function switch	0: close API status monitoring 7: open API status monitoring
[autoAnswer]	string	Automatic answering switch for extension. (Note: This parameter is valid only for IP extension and requires extension support)	Yes or no. The default value is no

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <ext id="208">
        <id>208</id>
        <staffid>1304081</staffid>
        <mobile>18603752800</mobile>
        <group>1</group>
        <group>2</group>
        <voicefile>busy</voicefile>
        <email>admin@hotmail.com</email>
        <Call_Restriction>3</Call_Restriction>
        <Call_Pickup>no</Call_Pickup>
        <No_Disturb>off</ No_Disturb>
        <Fwd_Type>0</Fwd_Type>
        <Fwd_Number>18603752801</Fwd_Number>
        <Fork>18603752802</Fork>
        <Off_Line_Num>218</Off_Line_Num>
        <WAKEUP>06:30</WAKEUP>
        <record>on</record>
        <autoAnswer>yes </autoAnswer>
        <api>7</api>
    </ext>
</Status>
```

Parameter description

Neglect, please refer to the parameter descriptions in the API response message of the querying extension.

(**Note:** Hyperlink can be returned with the shortcut of Alt +←)

Assign Group

The API is used to configure the parameters of an extension group and return the configuration results and status.

Configuration parameters include extension members, voice files, call distribution rules and so on.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Assign">
    <group id="1">
        <voicefile>NewMorning</voicefile>
        <distribution>sequential</distribution>
        <ext>200</ext>
        <ext>208</ext>
    </group>
</Control>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Parameter description	Parameter value specification
<group id>	int	Extension group number	1~20
[distribution]	string	Call allocation rule	Effective value: sequential, circular, simutaneous Default: circular
[ext]	string	Extension number, the group member	Must be valid extension

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <group id="1">
        <voicefile>NewMorning</voicefile>
        <ext id="200"/>
        <ext id="208"/>
         <!-- Empty Waiting List -->
     </group>
</Status>
```

Explanation: among them, <!-- Empty Waiting List --> indicates that there is no call waiting in the queue.

Particular attention:

1) After the OM system restarts, the extension in group will be cleared, and the group must be reconfigured. When the system restarts, OM pushes the BOOTUP event to the application server and you can judge whether the system restarted or not based on this even.

2) When the group is configured, the previous extensions of the group are still exist. So it is recommended that you should empty the group before configuring it.

Parameter description

Neglect, please refer to the parameter descriptions in the API response message of the querying group.

(**Note:** Hyperlink can be returned with the shortcut of Alt +←)

Assign voice menu

The API is used to configure parameters of a voice menu, and return the configuration results and status.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Assign">
    <menu id="1">
        <voicefile>welcome</voicefile>
        <repeat>3</repeat>
        <infolength>5</infolength>
        <exit>#</exit>
    </menu>
</Control>
```

Parameter description

Description: < > indicates imperative option, [] indicates option (when an argument is default value or null value, response message may not carry the parameters), | indicates perhaps.

Parameter name	Type	Parameter description	Parameter value specification
<menu id>	int	Voice menu number	1~50
[voicefile]	string	Voice file. The voice file is broadcast when a call is transferred to voice menu.	Only the DAT and PCM formats are supported, and the remote download path can be specified.
[repeat]	int	The times a voice file is played.	The valid value is 0~65535, 0 express loop play, and the default value is 0
[infolength]	int	The key check length. When the length of the key entered to the menu is transferred to the menu, the OM pushes the detected keystroke information to the application server.	The legal value is 1~255 (a byte length). The default value is 1
[exit]	char	The keystroke check terminator. When the caller transferred to the menu presses the exit key, the OM immediately pushes the detected keystroke information to the application server.	Legal values: 1~9, A~D, *, #. Default is empty.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Status>
    <menu id="1">
        <voicefile> welcome </voicefile>
        <repeat>3</repeat>
        <infolength>5</infolength>
        <exit>#</exit>
        <outer id="44" from="200" to="02167103750" trunk="02161208234"/>
        <visitor id="46" from="13012345678" to="02161204000"/>
    </menu>
</Status>
```

Explanation: The outer and visitor are respectively the outbound and incoming calls resided in the voice menu currently.

Note: If no call was diverted to the menu, the response body will carry the XML annotation fields: <! -- Empty Waiting List -- >.

Parameter description

Neglect, please refer to the parameter descriptions in the API response message of the querying menu.

(**Note:** Hyperlink can be returned with the shortcut of Alt +←)

2.1.3 Hold/Unhold

During the call, you can hold the call through the Hold command and release it through the Unhold command.

Hold

The API is used to keep the current call of an extension. After the other party is holded, the extension can initiate a new call via API.

If the holded call need to be resumed, can use Unhold command. In addition, after the call, the original call will be taken back automatically.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Hold">
    <ext id="200"/>
</Control>
```

Parameter description

Parameter Name	Type	Parameter description
<ext id>	string	Extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- The body is empty -->
```

Unhold

The API can be used to recover a holded cal.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Unhold">
    <ext id="200"/>
</Control>
```

Parameter description

Parameter Name	Type	Parameter description
<ext id>	string	Extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
```

```
Connection: close
Content-Length: 0
<!--the body is empty-->
```

2.1.4 Mute/Unmute

If you do not want the other party to hear your voice during the call, you can mute the other party by Mute command. And you can use the Unmute command to make the call return to normal.

Mute

During the extension call, this API can be used to make its microphone mute. That is, the other party can not hear the extension’s voice, but the extension can hear the other party's voice.

If you want to relieve the mute state, you can use the Unmute command to relieve it. And after the call, the mute state will be relieved automatically.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Mute">
      <ext id="200"/>
</Control>
```

Parameter description

Parameter name	**Type**	**Parameter description**
<ext id>	string	Extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
```

```
Connection: close
Content-Length: 0
<!-- the body is empty -->
```

Unmute

The API is used to resume the extension from mute state to the normal, to make the other party hear the extension's sound again.

If the Unmute is not executed after the Mute command is executed, at the end of the call, it will be cancelled automatically, but not affecting the next call.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Unmute">
    <ext id="200"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- the body is empty -->
```

2.1.5 Monitor

The API is used to implement an extension to listen to the current call of another extension.

Operation procedure

1. Extension A is currently idle, extension B is on the talking.
2. Excute the API of extension A listening to extension B.
3. After the API is executed successfully, extensions A ring, and than monitor the conversation between B and the other party after offhook.

Attention

- The monitor must open the "monitor" switch.
- The monitored and its calling party (another extension) must close the switch of "banned being monitored".
- One extension can be monitored by at most one extension.
- If the extension is in three party call, it will not be listened.
- The monitor range: extensions monitor only each other on the same OM device, not support multi-site equipment.
- The monitoring party onhooks but not affect the original call.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Monitor">
    <ext id="200"/>
    <ext id="208"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number. The first one is the monitoring party, and the second one is the monitored party.

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
```

```
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- the body is empty-->
```

Changing from monitor to interjection（Talk）

The API is used to implement an extension changing from monitor to interjection state.

Limitation conditions

Version requirements: Rev2.1.5.119 and above

Operation procedure

1. Extension A is listening to the call between extension B and its calling party.
2. Excute the API of changing from monitor to interjection.
3. After the API is executed successfully, extension A sets up the call with extension B, and the original calling party is holded.

Request example

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Talk">
    <ext id="201"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Sponsor extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
```

```
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- the body is empty-->
```

Changing from interjection to monitor（Listen）

The API is used to implement an extension changing from interjection to monitor state.

Limitation conditions

Version requirements: Rev2.1.5.119 and above

Operation procedure

1. Extension A is talking to extension B and its calling party.
2. Excute the API of changing from interjection to monitor.
3. After the API is executed successfully, extension A monitors the calls between extension B and its original calling party.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Listen">
    <ext id="201"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Sponsor extension number

Parameter description

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
```

```
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- the body is empty-->
```

2.1.6 Bargein

The API is used to force an extension into the call of another extension, thus forming tripartite call.

Operation procedure

1. Extension A is currently idle, extension B is busy.
2. Execute the API of extension A bargeining for extension B.
3. After the API is executed successfully, extensions A ring, offhook and than form tripartite call.

Attention

- The barge-in party must open the "barge" switch.
- The barge-in party and the call party (another extension) must be closed the switch of"block from being barge-in”.
- If the extension is in a tripartite meeting, it will not be barged in.
- The barge-in range: extensions must be on the same OM device, not supporting multi-site equipment. If the sponsor onhooks, the other two phone calls will be hanged up.
- After the barging party exits from tripartite call, it can barge again. For barged party and its call party, they can not join the call again after exiting.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Bargein">
    <ext id="200"/>
    <ext id="208"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number. The first one is the barging party, and the second is the barged party

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!-- the body is empty-->
```

2.1.7 Clear

This type of API is used to force to hang up the specified call on OM. The objects which can be hanged up include extension, incoming call and outbound call.

Rule description

- For the extension, the judge demolition is ext id.
- For incoming and outbound calls, the judge demolitions are visitor id and outer id.
- For the two sides of a call, no matter which side is cleared, the call will be released.

Request example

1）Clear extension:

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Clear">
    <ext id="200"/>
</Control>
```

2）Clear incoming call:

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Clear">
      <visitor id="2"/>
</Control>
```

3）Clear outbound call:

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Clear">
      <outer id="20"/>
</Control>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	String	Extension number. All extension numbers and their corresponding line numbers can be obtained by querying the device information.
<visitor id>	int	Incoming call identification
<outer id>	int	Outbound call identification

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

2.2 Transfer Command

The application server can send transfer commands to the OM to implement calling, transfering, queue, and tripartite conference.

2.2.1 Connect command

The connect command can realize the calls, such as extension calling extention, voice menu calling outer call, two-way outbound call and transferring for the incoming or outbound calls, etc.

Extension calling extension

This API is used for initiating a call from one extension to another.

The API is used for allowing an extension to initiate a call to another extension, so that both can establish a call.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="200"/>
    <ext id="208"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number. The first one is the calling extension, and the other one is the called extension. Whether the OM call the calling extention or the called extention depends on the parameter of API_CALLING.

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
```

```
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

Extension calling external phone

The API is used for initiating a call from an extension to an external call.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="208"/>
    <outer to="13012345678"/>
</Transfer>
```

Specify trunk to call outer:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="208"/>
    <trunk id="50017"/>
    <outer to="13012345678"/>
</Transfer>
```

Add prefix to call outer:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="208"/>
    <outer to="9,13012345678"/>
</Transfer>
```

Multiple dial：

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <ext id="208"/>
    <outer to="13012345678,,1"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number
<outer to>	string	The identification number of an outbound call
[trunk id]	string	Specified trunk number, can be IP or analog trunk number

Note:

- Number 9 from the command of calling outer by adding prefix is prefix. By this command, you can call outer specifying IP or analog trunk. However, note that the dialing rule for the device should be "Outbound dialing with prefix".
- From the command of multiple dial, “,,1”means that the caller presses the button 1. You can continue add “,” behind the symbol “,,” to delay the key time, a comma indicates a delay for one second.

Response example

The outbound response by IP trunk:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <outer id="66" from="208" to="13012345678" trunk="50017" callid="8258"/>
    <ext id="208"/>
</Event>
```

The outbound response by analog trunk

Note: When call outer via analog, there are no “**outer id**、**trunk**、**callid**” in the response information. But then OM pushes the "**TRANSIENT**" event message to the application server again, which adds the full amount of the parameter information.

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
```

```
    <outer from="208" to="13012345678"/>
    <ext id="208"/>
</Event>
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying the outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring incoming call to extension

The API is used for initiating a call from an incoming call to an extension.

Prompt tone description

- Before OM transfers the incoming call to the extension, it will first play the forwarding tone "XXX is transferring, please hold on." And if the extension is configured with the staff number, XXX is the staff number, otherwise is the extension number.
- If the transfer prompt tone in OM can not meet the requirements, you can specify a new voicefile temporarily, the voicefile plays only in this transfer operation.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="14"/>
    <ext id="200"/>
</Transfer>
```

Specify a temporary voice file:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="14"/>
    <ext id="200"/>
    <voicefile>200+NewMorning</voicefile>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	Incoming call number
<ext id>	string	Extension number
[voicefile]	string	Voice file, used to temporarily replace the prompt tone, supports digital string and DAT, PCM format. If there are multiple files, plus signs can be used to connect them.

Note:

The number of voice files specified in <voicefile> is less than or equal to 10. As long as the voice file exists (available for download), it can be played properly.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <visitor id="14" from="13012345678" to="02161208234" callid="49226" />
    <ext id="200" />
</Event>
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying the incoming call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring incoming call to external phone

The API is used for initiating a call from an incoming call to an external call.

Execution Authority

Perform outbound call command before, you need to open the authority switch of the external connecting external (Id is 1070).

Prompt tone explanation

- By default, there is no forwarding tone when the call is transferred outer.

- If you need transfer prompt, like transferring extention, you can specify the voice file in the API (Valid only in this call).

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="3" />
    <outer to="13012345678"/>
</Transfer>
```

Display calling number:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="3" />
    <outer to="13012345678" display="18201008888" />
</Transfer>
```

Note:

For the command of specifying trunk or adding prefix, you can refer to the command of Extension calling external phone. And for the command of playing voice file temporarily, you can refer to Transferring incoming call to extension.

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	The identification number of an incoming call
<outer to>	string	External phone number
[display]	string	Incoming call number for displaying. **Note:** When an incoming call is relayed, you need to get the calling number and add it to the display field in real time.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <visitor id="46" from="13012345678" to="02161204000" callid="41006"/>
    <outer to="02167103750"/>
</Event>
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying the incoming call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring incoming call to menu

The API is used for transfering an incoming call to a voice menu (menu), thus realizing:

- Play voice files to the caller. Among them, the broadcast times and contents of voice files can be controlled.
- Button information entered by caller can be checked. When the button meets the key reporting condition of the menu configuration, OM reports the key information to the application server.

Playing temporary voice files

- If a new voice file is specified temporarily when the call forwarding menu is called, the OM no longer broadcasts the voice file configured on the menu, but broadcasts the voice file specified temporarily, and plays only once. After the voice file is played, OM will report the EndOfAnn event to the application server.
- But, the reporting rules for the DTMF event still follow the configuration of menu.

Examples of Playing temporary voice files:

- **Key error correction prompt:** When the pressed key in a menu is out of order, the call can be re-connected to the menu, playing specified correction note, instead of the voice file configured on the menu.

- **Broadcast dynamic voice:** Such as balances, work orders and other unfixed contents.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="41" />
    <menu id="1"/>
</Transfer>
```

Note: For the command of playing voice file temporarily, you can refer to Transferring incoming call to extension.

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	The identification number of an incoming call
<menu id>	int	voice menu number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <visitor id="41" from="02167103750" to="02161204000" callid="41001"/>
    <menu id="1" />
</Event>
```

Parameter description

Neglect, please refer to the parameter description of the API response message for querying the incoming call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring outbound call to extension

The API is used for transfering an outbound call to an extension.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <outer id="14"/>
    <ext id="200"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<outer id>	int	The identification number of an outbound call
<ext id>	string	Extension number

Response example

```
<Event attribute="TRANSIENT">
    <outer id="14" from="200" to="13012345678" trunk="61208111" callid="8258"/>
</Event>
```

Parameter description

Neglect, please refer to the parameter description of the API response message for querying the outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring outbound call to external phone

The API is used for transferring an outbound call to an external call.

Execution authority

- Perform outbound call command before, you need to open the authority switch of the external connecting external (Id is 1070).

Prompt tone explanation

- By default, there is no forwarding tone when the call is transferred outer.

- If you need transfer prompt, like transferring extention, you can specify the voice file in the API (Valid only in this call).

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <outer id="14" />
    <outer to="13012345678"/>
</Transfer>
```

Note:

For the command of specifying trunk or adding prefix, you can refer to the command of Extension call external phone. And for the command of playing voice file temporarily, you can refer to Transferring incoming call to extension.

Parameter description

Parameter name	Type	Parameter description
<outer id>	int	The identification number of an outbound call
<outer to>	string	External phone number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
      <outer id="14" from="200" to="02167103750" trunk="02161204000" callid="49174"
/>
      <outer to="13012345678"/>
</Event>
```

Parameter description

Neglect, please refer to the parameter description of the API response message for querying the outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Transferring outbound call to menu

The API is used for transferring an outbound call to a voice menu, thus achieving:

- Play voice files to the caller. Among them, the broadcast times and contents of voice files can be controlled.

- Can check button information entered by caller. After the button meets the key reporting condition of the menu configuration, OM reports the key information to the application server.

Playing voice files temporarily

- If a new voicefile is specified temporarily when the call is transfered to menu, the OM no longer broadcasts the voice file configured by menu, but plays another one designated temporarily, and the voice file plays only once. After that OM reports the EndOfAnn event to the application server.

- Even so, the DTMF reporting rules still follow the menu configuration。

Examples of extended function applications:

- **Key error correction prompt:** When the pressed key in a menu is out of order, the call can be re-connected to the menu, playing the specified correction note, instead of the voice file configured on the menu.
- **Broadcast dynamic voice:** Such as balances, work orders and other unfixed contents.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
      <outer id="41" />
      <menu id="1"/>
</Transfer>
```

Note: For the command of playing voice file temporarily, you can refer to Transferring incoming call to extension。

Parameter description

Parameter name	Type	Parameter description
<outer id>	int	The identification number of an outbound call
<menu id>	int	Voice menu number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
      <outer id="14" from="200" to="02167103750" trunk="02161204000"
callid="49174" />
      <menu id="1"/>
</Event >
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying the outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Menu calling extension

The API is used for initiating a call from a voice menu to an extension, thus realizing:

- Play voice files to the extension. Among them, the broadcast times and contents of voice files can be controlled.
- Button information entered by extension can be checked. When the button meets the key reporting condition of the menu configuration, OM reports the key information to the application server.

Common usage scenarios

- Wake up service of the hotel.
- Meeting notice, etc.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <menu id="1"/>
    <ext id="200"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<menu id>	int	Voice menu number
<ext id>	string	Extension number

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
```

```
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

Menu calling external phone

The API is used for initiating a call from a voice menu to an external call, thus realizing:

- Play voice files to the external phone. Among them, the broadcast times and contents of voice files can be controlled.
- Button information entered by external call can be checked. When the button meets the key reporting condition of the menu configuration, the OM reports the key information to the application server.

Execution authority

- Perform outbound call command before, you need to open the authority switch of the external connecting external (Id is 1070).

Common usage scenarios

- Customer service quality score.
- Voice navigation and user survey.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <menu id="1"/>
    <outer to="13012345678"/>
</Transfer>
```

Note:

For the command of specifying trunk or adding prefix,you can refer to the command of Extension calling external phone.

Parameter description

Parameter name	Type	Parameter description
<menu id>	int	Voice menu number
<outer to>	string	Outbound call number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <outer id="1" from="" to="13012345678" trunk="02161208234" callid="20481"/>
    <menu id="1"/>
</Event>
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying the outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Two-way outbound call (call-back)

The API is used for initiating a call from an external call to another.

Call order

First, call the first number, then call the second number after the caller offhooks and hears the ring-back tone.

Trunk resource requirements

- There are at least two idle available trunk resources.
- The first call is from IP trunk by default, it can be adjusted to analog trunk by the parameter of CB_TYPE.
- The second outbound call follows the outbound dialing rules configured on OM.

Execution Authority

Perform outbound call command before, you need to open the authority switch of the external connecting external (Id is 1070).

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <outer to="13012345678"/>
    <outer to="02167103750"/>
</Transfer>
```

Specify trunk to call outer:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
     <outer to="13012345678" display="50018"/>
     <outer to="02167103750" display="50019"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<outer to>	string	External phone number. The first one is the caller, the second one is the called party.
<display>	string	Specified trunk number, can be IP/analog trunk. The parameter API_DISPLAY and API_CPN should be configured.

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
     <outer id="3" from="" to="13012345678" trunk="02161208234" callid="45059"/>
</Event>
```

Parameter description

Neglect , please refer to the parameter description of the API response message for querying outbound call.

(**Tips:** shortcut links are returned with Alt + ←)

Voice insertion

The API is used to insert speech into the party that is calling. After the insertion command is executed, the voice is heard by the calling party, and the calling party is mute, and the original call is automatically restored after the voice playback is finished.

If a two-way slot is needed, two orders can be executed simultaneously.

Limiting condition

Version: Rev, 2.1.5.117 and above

Suit scene

Include three scenarios of extension calling extension, transferring incoming call to extension and extension calling outbound call.

Request example

Insert voice into extension:

```
<?xml version="1.0" encoding="udf-8" ?>
<Transfer attribute="Connect">
      <ext id="200" />
      <voicefile>welcome</voicefile>
</Transfer>
```

Insert voice into incoming call:

```
<?xml version="1.0" encoding="udf-8" ?>
<Transfer attribute="Connect">
      <visitor id="15" />
      <voicefile>welcome</voicefile>
</Transfer>
```

Insert voice into outbound call:

```
<?xml version="1.0" encoding="udf-8" ?>
<Transfer attribute="Connect">
      <outer id="22" />
      <voicefile>welcome</voicefile>
</Transfer>
```

Parameter description:

Parameter name	Type	Parameter description
<ext id>	string	Extension number
<visitor id>	int	The identification number of an incoming call
<outer id>	int	The identification number of an outbound call
<voicefile>	string	Voice file

Response example

```
HTTP/1.0 200 OK
Date: Tue, 27 Jun 2017 02:51:12 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

2.2.2 Queue Command

The API is used for transferring the calls to the queue, so that the caller can wait in line when the call is busy until the called party may accept the call.

Queue Type

Currently, there are only two objects supporting queue function, respectively:

- Extension queue;
- Extension group queue.

Transferring incoming call to extension queue

The API is used for transfering a call to an extension queue. When some extension is free, OM assigns the calls to the extension in turn.

That is:

- If the extension is free, the incoming call will be forwarded directly to the extension;
- If the extension is busy, the caller waits in the queue and listens to the waiting music. When the extension is free, the first caller of the queue is forwarded to the extension.

Note:

The waiting music needs to be configured on extensions. Otherwise, the system plays busy tone automatically when the caller is in the queue (For the previous version of 2.1.5.114, play mute tone).

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Queue">
   <visitor id="1"/>
   <ext id="200"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	The identification number of an incoming call
<ext id>	string	Extension number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
      <visitor id="1" from="02167103750" to="02161208234" callid="16385"/>
      <ext id="200"/>
      <state>wait</state>
</Event>
```

Parameter Description: <state>wait</state> is optional field. When this field exists, it means the extension is busy at the moment. By contrary, it means the extension is idle, and the incoming call has been transferred to the extension.

Parameter description

Neglect, please refer to the parameter description of the API response message for querying incoming call.

(**Tips:** shortcut links back to Alt + ←)

Transferring incoming call to extension group queue

The API is used for transfering the call to the extension group. The OM assigns the call to one or some free extension in the group in accordance with the call distribution rule of the extension group, thus establishing call between the incoming call and some extension of the group.

Busy/No response

- The extension group has queuing function. If no extension can accept the incoming call (such as busy, hang or don't disturb configured), the call will be waiting in line, and listen to the waiting music configured on extension, until some extension become idle.

- If there are idle extensions can accept the incoming call, the extension rings first, and then hang up to establish a call. If the extension is not answered promptly, the call will automatically jump to the next free extension.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Queue">
   <visitor id="1"/>
   <group id="1"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	The identification number of an incoming call
<group id>	int	Extension group number

Response example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
```

```
    <visitor id="1" from="13012345678" to="02161208234" callid="20481"/>
    <group id="1"/>
</Event>
```

Parameter description

Neglect, please refer to the parameter description of the API response message for querying the incoming call.

(**Tips:** shortcut links are returned with Alt + ←)

2.2.3 Conference command

The API is used to initiate meeting and currently supports only three party meeting.

Operation procedure

1. Extension A is talking to B.
2. The extension A keeps the original call holding.
3. Extension A initiates a new call to C.
4. At this point, you can use the API to implement the three party meeting of A, B, and C with the extension A as the host.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Conference">
    <ext id="200"/>
</Transfer>
```

Parameter description

Parameter name	Type	Parameter description
<ext id>	string	Extension number, the number of conference sponsor

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
```

```
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

2.3 Accept command

If the API function switch is configured as mode of pre-answer control for incoming call, When the phone calls the trunk, OM will push the API report message (the INVITE event) to the application server, and wait the application server to control the incoming call within a certain period of time (Five seconds by default).

The application server can control the incoming call as follows:

- **Accept:** If the call is expected to receive, the application server could send the API of Accept command, and OM will allow the call to go into the follow-up process.
- **Clear:** If the call is expected to reject, the application server could send the API of Clear command, and OM will hang up the call directly.

Timeout processing mechanism:

If the application server does not control the call within the specified time, OM accepts the call by default, that is the incoming call is accepted.

Request example

```
<?xml version="1.0" encoding="utf-8" ?>
<Notify attribute="Accept">
    <visitor id="1"/>
</Notify>
```

Parameter description

Parameter name	Type	Parameter description
<visitor id>	int	The identification number of an incoming call

Response example

```
HTTP/1.0 200 OK
Date: Wed, 22 Feb 2017 08:04:44 GMT
Server:
X-Frame-Options:DENY
Cache-Control:no-cache
Pragma:no-cache
Expires:-1
Access-Control-Allow-Headers: accept, Content-Type
Access-Control-Allow-Origin: (null)
Connection: close
Content-Length: 0
<!--the body is empty-->
```

2.4 Event report

When the OM system starts, configuration changes, extension status changes, or call status changes, the OM automatically triggers the corresponding event. You can listen to these events if your application server opens the listening port.

You can according to the corresponding event monitor some status of OM, extension, line, etc.

2.4.1 System event

Such event reports are mainly used for reporting changes of system status. When system status changes, OM pushes such event reports to the application server.

System startup event (BOOTUP)

When the system is started, OM pushes the report to the application server.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BOOTUP">
</Event>
```

OM configuration change event (CONFIG_CHANGE)

When the web page configuration changes, OM pushes the report to the application server, so that the application server can update and synchronize the OM configuration timely.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="CONFIG_CHANGE">
</Event>
```

2.4.2 Extension status change events

When the line status of an extension changes, the OM device pushes the report to the application server, so that the application server can monitor the status of the extension in real time.

There are four types of line states that can be reported by extension: IDLE, BUSY, ONLINE and OFFLINE.

Extension busy event (BUSY)

When extension’s status changes from idle to busy, the OM device pushes this event to the application server.

Note: IP extension will not immediately report the BUSY event when it hangs up, but wait until the call is dialed.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BUSY">
   <ext id="208" />
</Event>
```

Extension free event（IDLE）

When extension’s status changes from busy to idle, the OM device pushes the report to the application server.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="IDLE">
   <ext id="208" />
</Event>
```

Extension on-line event（ONLINE）

When IP extension’s status changes from offline to online, or when the address of the IP extension changes, the OM device pushes this event report to the application server.

Note: Only the IP extension has this event, and the analog extension has not.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ONLINE" RegisterIP="10.129.91.202" >
   <ext id="208" />
</Event>
```

Extension offline event（OFFLINE）

When IP extension’s status changes from on-line to offline, the OM device pushes this report to the application server.

Note: Only the IP extension has this event, and the analog extension is not.

There are situations where OFFLINE events can be triggered:

- IP extension logout;
- IP extension is not registered during the cycle refresh registration because of disconnecting the network, power outages or other reasons.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="OFFLINE">
   <ext id="208" />
</Event>
```

2.4.3 Call status change events

When the call status of the extension or trunk changes, OM pushes the event report to the application server.

Ringing event (RING)

When an extension rings, OM pushes the event report to the application server.

Report example

- **Transfer an incoming call to an extension, and the extension rings:**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="RING">
   <ext id="200" />
   <visitor id="22" from="13012345678" to="02161208234" callid="16408" />
</Event>
```

- **An extension call another extension, and the called extension rings:**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="RING ">
   <ext id="200" />
   <ext id="208" />
</Event>
```

- **When an extension calls an external phone through the API, and also the calling mode is “Call the called phone at first, call the caller after the called phone rings”, the caller rings:**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="RING ">
   <ext id="200" />
   <outer id="22" from="200" to="13012345678" trunk="02161208234" callid="16406" />
</Event>
```

- **When an extension calls an external phone or an extension calls another extension through the API, and also the calling mode is ”Call the caller at first, call the called phone after the caller hangs up”, the caller rings:**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="RING ">
   <ext id="200" />
   <visitor from="" />
</Event>
```

- **Menu calls extension, and the extension rings:**

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="RING">
   <ext id="200" />
```

```
  <menu id="1" />
</Event>
```

ALERT event (ALERT)

When the called phone rings, it will give an alert message to the caller, thus OM pushes the event report to the application server.

Report example

- **Transfer an incoming call to an extension, and the extension alerts:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ALERT">
  <visitor id="22" from="13012345678" to="02161208234" callid="16406" />
  <ext id="200" />
</Event>
```

- **An extension calls an external phone, the external phone alerts:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ALERT">
  <ext id="200" />
  <outer to="13012345678" />
</Event>
```

- **An extension calls another extension, and the called extension alerts:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ALERT">
  <ext id="200" />
  <ext id="208" />
</Event>
```

- **Menu calls external phone, the external phone alerts:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ALERT">
  <outer id="22" from="" to="13012345678" trunk="02161208234" callid="16406" />
</Event>
```

Call response event (ANSWER)

When the extension hangs up to answer the call, OM pushes the event report to the application server.

Report example

- **An extension calls another extension, the called extension picks up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ANSWER">
   <ext id="200" />
   <ext id="208" />
</Event>
```

- **Transfer an incoming call to an extension, the extension picks up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ANSWER">
   <ext id="200" />
   <visitor id="31" from="13012345678" to="02161208234" callid="16417" />
</Event>
```

Answered event (ANSWERED)

When the called phone is checked the off hook, OM pushes this event report to the application server.

Report example

- **When an extension calls an external phone, the extension checks the external phone picks up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ANSWERED">
   <outer id="34" from="200" to="13012345678" trunk="02161208234" callid="16418" />
   <ext id="200" />
</Event>
```

- **When an extension calls another extension, the caller checks the called extension picks up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ANSWERED">
   <ext id="200" />
   <ext id="208" />
</Event>
```

End of call event (BYE)

When a call is released, the OM pushes the event report to the application server.

Report example

- **When the call between the incoming call and extension is over, the incoming call has been hung up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BYE">
   <visitor id="42" from="13012345678" to="02161208234" callid="16426" />
   <ext id="200" />
</Event>
```

- **When the call between an extension and an outbound call is over, the extension has been hung up:**
    - Type 1:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BYE">
<ext id="200" />
<outer to="13012345678" />
</Event>
```

- Type 2:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BYE">
<outer id="38" from="200" to="13012345678" trunk="02161208234" callid="16422" />
</Event>
```

- **When the call between the incoming call and outbound call is over, the incoming call has been hung up:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BYE">
  <visitor id="50" from="02167103750" to="02161204000" callid="49202" />
  <outer id="51" from="200" to="13012345678" trunk="02161208234" callid="49202" />
</Event>
```

- **When the two-way outbound call is over, each of the two outbound calls has a BYE event:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="BYE">
  <outer id="48" from="" to="13012345678" trunk="28900928" callid="49200" />
</Event>
```

Call transfer event (DIVERT)

When a call is transferred or redirected, the OM pushes this event report to the application server.

Report example

- **When the incoming call 13012345678 calls an extension, the call is transferred because of the call transfer configured on the extension:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DIVERT">
  <visitor id="5" from="13012345678" to="204" callid="12293" />
</Event>
```

- **When the extension 208 calls another extension, the call is transferred because of the call transfer configured on the extension:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DIVERT">
  <ext id="208" />
</Event>
```

Call temporary event（TRANSIENT）

This event is usually used as a response message for the Transfer request, and in particular OM also pushes the event message.

In what case does OM push the event:

When an extension calls an external phone through the analog trunk, OM can not immediately provide outer id, trunk, callid, so it will push this event after the call is sent outer through anolog trunk to supplement the outbound call information.

Note: The IP trunk has no this phenomenon.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
    <outer id="66" from="208" to="13012345678" trunk="61208111" callid="8258"/>
    <ext id="208"/>
</Event>
```

Call failure event（FAILED）

This event usually appears as a response message for the API request, and in particular the OM also pushes the event message.

In what case does OM push the event:

- This event is triggered when a call is initiated using OM API but fails during execution process.
- The event is triggered only in extreme cases when the call is initiated by manual dialing.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="FAILED">
    <called id=" number "…… />
    <err code="number"   reason="failed reason " />
</Event>
```

Error code description

Error code field	Description
<err code="1" reason="temp failure" />	Fail

Error code field	Description
<err code="2" reason="restricted" />	The calling rights of the caller is limited.
<err code="3" reason=" offline " />	The called extension is not online.
<err code="4" reason="other call" />	The called extension is currently talking to another terminal.
<err code="5" reason="no circuit" />	Trunk resources are inadequate, unable to execute the call .
<err code="6" reason="busy" />	The called extension is busy.
<err code="7" reason="not exist" />	Extension does not exist.
<err code="8" reason="offline" />	The calling extension itself is not online.

Report example

- **IP extension offline:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="FAILED">
   <ext id="208" />
   <err code="8" reason="offline" />
</Event>
```

- **Extension 208 is making other call:**

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="FAILED">
   <ext id="200" />
   <ext id="208" />
   <err code="4" reason="other call" />
</Event>
```

2.4.4 Incoming call control process events

The event is pushed only when the API call control switch is turned on and there is an incoming call.

Handing process of incoming call:

1. Incoming call calls the trunk number;
2. OM determines whether the switch of “Pre-answer control for incoming call” is open;
3. OM determines whether the switch of “Post-answer control for incoming” is open;
4. Enter into subsequent processing flow.

Control switch of pre-answer control for incoming call:

- If the switch is turned on, when a caller calls corresponding trunk, OM will not immediately accept the call, but pushes the event of describing the phenomenon (INVITE event) to the application server. If the application server expects the OM to accept the call, will execute the Accept API request. If it wants OM to deny the call, will execute the Clear API request. If the application server has not received the INVITE event or received the event but did not excute some control operation since the network or system failure occurred, the OM will automatically receive the call after waiting for a timeout. (Note: The time is controlled by the parameter of API_RESPONSE_TO, 5 seconds by default, configuration range for 1-120s, and no more than 30s proposed)
- If the switch is closed, when a call calls the trunk, the OM automatically accepts the call and does not push the INVITE event to the application server.

Control switch of post-answer control for incoming:

- If the switch is turned on, when a call calls the trunk, OM will immediately accept the call, put it on and mute it, meanwhile push the event of describing the phenomenon (INCOMING event) to the application server. The application server can execute the Transfer API request to transfer the call to the voice menu, extension or extension group, etc. If the application server has not received the INCOMING event or received the event but doesn’t excuted some control operation since the network or system failure occurred, the OM will automatically transfer the call to the operator after waiting for a timeout. (Note: The time is controlled by the parameter of API_RESPONSE_TO, 5 seconds by default, configuration range for 1-120s, and no more than 30s proposed)

  warm Tip: In order to prevent the application server abnormalities, we have some changes on Rev2.1.5.112 and later versions, supporting the mechanism of transferring operator after a timeout. For the previous version, OM still not process the incoming call after a timeout, and the caller will remain mute until the phone hang up.

- If the switch is closed, OM will automatically arrange transfering calls or navigation according to the configuration, such as playing voice about welcome. Also, it will not push the INCOMING event anymore.

Incoming call request event (INVITE)

In the case that the trunk's " **Pre-answer control for incoming** " switch is turned on, the OM pushes the INVITE event to the application server when a call calls the trunk.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="INVITE">
    <trunk id="02161204000" />
    <visitor id="31" from="13012345678" to="02161204000" callid="36895" />
</Event>
```

Incoming call event (INCOMING)

In the case that the trunk's " **Post-answer control for incoming** " switch is turned on, when the caller calls the trunk, OM will push the INCOMING event to the application server after the call is answered.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="INCOMING">
    <trunk id="02161204000" />
    <visitor id="31" from="13012345678" to="02161204000" callid="36895" />
</Event>
```

2.4.5 Key information event (DTMF)

This event is used to report the key information entered by the user.

After the call (incoming call/outbound call/extension) is transferred to the voice menu, when the user presses the button and the key information satisfies the configuration condition of the menu, the OM pushes the event to the application server.

Description: OM doesn’t report DTMF event when you press the keys by the multi-level voice navigation (IVR) configured on web page, but when you press the keys of *66 or *67 to binding or unbunding, the DTMF event will be reported.

Report example

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="4" from="13012345678" to="02161208234" callid="36868">
        <info>12345</info>
        <menu id="1" />
    </visitor>
</Event>
```

Parameter description

Parameter name	Parameter description
<visitor | outer>	Incoming call or outbound call
<id>	The identification number of an incoming call or outbound call, can be used to transfer the call
<from>	The calling party number
<to>	The called phone number
<callid>	The relative unique number of the call
[trunk]	Trunk number
<info>	User key information
[menu]	A voice menu for incoming or outbound calls

More examples

Situation 1: when an incoming call is connected to a voice menu and the dial length reaches the dial length of the menu, the OM pushes the report to the application server. (Menu 1 configured: dial detection length as 5, key terminator as #, key information as 12345)

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="4" from="13012345678" to="02161208234" callid="36868">
        <info>12345</info>
        <menu id="1" />
    </visitor>
</Event>
```

Situation 2: When an incoming call is connected to a voice menu, and the terminator is pressed, the OM pushes the report to the application server. (Menu 1 configured: dial detection length as 5, key terminator as #, key information as 200#).

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="4" from="13012345678" to="02161208234" callid="36868">
        <info>200#</info>
        <menu id="1" />
    </visitor>
</Event>
```

Situation 3: When the outbound call is transferred to the voice menu, and the dial length of the user reaches the dial length of the menu, the OM pushes the report to the application server. (Menu 1 configured: dial detection length as 5, key terminator as #, key information as 12345).

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <outer id="4" from="200" to="13012345678" callid="68671" trunk="02161208234">
        <info>12345</info>
        <menu id="1" />
    </outer>
</Event>
```

2.4.6 The event of the voice file playing into end (EndOfAnn)

When a call is transferred to a voice menu, the OM broadcasts the voice file to the calling party. At the end of the broadcast of the voice file, OM pushes the event to the application server.

Note: During the voice file plays, if the user presses the button, the voice stops playing, and does not push the EndOfAnn event.

Report example

Transferring incoming call to menu:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<Event attribute="EndOfAnn">
    <visitor id="67" from="13012345678" to="208" callid="32835" />
    <menu id="1" />
</Event>
```

Transferring outbound call to menu：

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="EndOfAnn">
    <outer id="4" from="200" to="13012345678" callid="68671" trunk="02161208234" />
    <menu id="1" />
</Event>
```

Transferring an extension to menu：

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="EndOfAnn">
    <ext id="200" />
    <menu id="1" />
</Event>
```

Report description

Parameter name	**Parameter description**
<visitor | outer>	Incoming call or outbound call
<id>	The identification number of an incoming call or outbound call, can be used to transfer the call.
<from>	The caller number
<to>	The called phone number
<callid>	The relative unique number of the call
[trunk]	Trunk number
<info>	User key information
<menu id>	A voice menu for incoming or outbound calls

2.5 Call record report (CDR)

(**Tips:** shortcut links are returned with Alt + ←)

A call recording report is a record and statistics report of OM on a call from start to finish. When a call is released, the OM pushes the report in real-time to the application server.

There are six different types of CDR, such as IN, OU, FI, FW, LO and CB, corresponding to different application scenarios.

CDR description

IN call record description:

- When the incoming call is finished, IN call record is generated;
- When the call of extension calling another extension is finished, the called extension generates IN call record. (Note: The calling extension generates Lo call record)

LO call record description:

- After the call of extension calling extension, the calling extension generates LO call record.
- An incoming call is relayed to an extension via analog trunk. At the end of the call, the analog trunk generates LO call record.

OU call record description:

After the outbound call is finished, OU call record is generated.

FI call record description:

- When the extension transfers its call to another phone (extension or external phone), after the call, FI call record is generated.

FW call record description:

- When the extension transfers its call to another phone (must be external phone), after the call, FW call record is generated.

CB call record description:

- After the two-way outbound call, OM pushes CB call record to the application server.

Report message format

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="value">
   <callid> value </callid>
   <visitor|outer id=" value " />
   <TimeStart> value </TimeStart>
   <Type> value </Type>
   <Route> value </Route>
   <CPN> value </CPN>
   <CDPN> value </CDPN>
   <TimeEnd> value </TimeEnd>
   <Duration> value </Duration>
   <TrunkNumber> value </TrunkNumber>
   <Recording> value </Recording>
   <RecCodec>PCMU</RecCodec>
</Cdr>
```

Parameter description

Parameter name	Type	Description
<Cdr id>	string	Call record id
<callid>	int	The relative unique number of the call
[visitor | outer id]	int	The identification number of an incoming call or outbound call, can be used to transfer the call
<type>	String	The type of call record, contains IN/OU/FI/FW/LO/CB
<Route>	String	IP (IP trunk) / XO (analog trunk) / IC (internal) / OP (Operator)
<TimeStart>	string	The time of initiating the call
<TimeEnd>	string	The time of ending the call
<CPN>	string	The calling number
<CDPN>	string	The called number

Parameter name	Type	Description
<Duration>	int	Call duration, if the value is 0, indicates that the call has not been connected.
<TrunkNumber>	string	Trunk number
[Recording]	string	The relative preservation path of a recording file, format: year month day/recording file. The format of the recording file name: 1. calling number_called number _year month day-hour minute second_relative unique identifier for call.wav 2. calling number_called number _year month day-hour minute second_relative unique identifier for call_cd.wav 3. calling number_called number _year month day-hour minute second_relative unique identifier for call_cg.wav **cg** indicates calling recording, **cd** indicates called recording.
[RecCodec]	string	Encoding mode, determine the format of recording files, values: G729, G711 (PCMA, PCMU). Note: Recording mode for local recording, G711 encoding will generate PCM format recording, OM devices automatically synthesize and storage WAV files, you can directly download and play them. But for G729 coded recordings generated by G729 encoding, OM does not synthesize recording files and only keeps DAT files in the direction of send and recv, which can't be broadcast directly, and needs to be locally synthesized after downloading.

2.5.1 IN

1. **When incoming calls are transferred via the IP trunk, 2 IN call records are generated.**

- **IN call record of the extension which acts as called party:** External phone 18201008888 calls the extension 200，the duration of the call is 2 seconds and the corresponding recording is that of the extension 200.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="2820170309162713-0">
   <callid>32818</callid>
   <visitor id="48" />
   <TimeStart>20170309162708</TimeStart>
   <Type>IN</Type>
   <Route>IP</Route>
   <CPN>18201008888</CPN>
   <CDPN>200</CDPN>
   <TimeEnd>20170309162713</TimeEnd>
   <Duration>2</Duration>
   <TrunkNumber>888</TrunkNumber>

<Recording>20170309/18201008888_200_20170309_162711_8032_cd.wav</Recor
ding>
   <RecCodec>G729</RecCodec>
</Cdr>
```

- **IN call record of the external phone which acts as called party:** External phone 18201008888 calls the trunk 888，the duration of the call is 17 seconds and the corresponding recording is that of the trunk 888.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="2920170309162714-0">
   <callid>32816</callid>
   <visitor id="48" />
   <TimeStart>20170309162657</TimeStart>
   <Type>IN</Type>
   <Route>IP</Route>
   <CPN>18201008888</CPN>
   <CDPN>888</CDPN>
   <TimeEnd>20170309162714</TimeEnd>
   <Duration>17</Duration>
   <TrunkNumber>888</TrunkNumber>
```

```
<Recording>20170309/18201008888_888_20170309_162657_8030_cd.wav</Recor
ding>
   <RecCodec>G729</RecCodec>
</Cdr>
```

2. **When incoming calls are transferred by analog trunk, 2 call records are generated, respectively as IN and LO (The LO call record can be filtered out).**

   ◦ **IN call record of the extension which acts as called party:** External phone 18201008888 calls the extension 200，the duration of the call is 4 seconds and the corresponding recording is that of the extension 200.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="3120170309162952-0">
   <callid>32820</callid>
   <visitor id="51" />
   <TimeStart>20170309162945</TimeStart>
   <Type>IN</Type>
   <Route>XO</Route>
   <CPN>18201008888</CPN>
   <CDPN>200</CDPN>
   <TimeEnd>20170309162952</TimeEnd>
   <Duration>4</Duration>
   <TrunkNumber>02161205555</TrunkNumber>

<Recording>20170309/18201008888_200_20170309_162948_8034_cd.wav</Recor
ding>
   <RecCodec>PCMU</RecCodec>
</Cdr>
```

   ◦ **LO call record of the external phone which acts as called party:** External phone 18201008888 calls the extension 200, the duration of the call is 6 seconds and the corresponding recording is that of the trunk 02161205555.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="5820170309175319-0">
<callid>32861</callid>
<visitor id="93" />
```

```
<TimeStart>20170309175311</TimeStart>
<Type>LO</Type>
<Route>IC</Route>
<CPN>18201008888</CPN>
<CDPN>200</CDPN>
<TimeEnd>20170309175319</TimeEnd>
<Duration>6</Duration>
<TrunkNumber></TrunkNumber>
<Recording>20170309/18201008888_02161205555_20170309_175313_805D_cg.w
av</Recording>
<RecCodec>PCMU</RecCodec>
</Cdr>
```

2.5.2 OU

1. **Outbound call produces only a single call record of OU.**

   - **OU call record of the extension which acts as the caller:** Extension 200 call the external phone 18201008888，the duration of the call is 12 seconds and the corresponding recording is that of the extension 200.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="2020170309161117-0">
  <callid>32801</callid>
  <outer id="34" />
  <TimeStart>20170309161101</TimeStart>
  <Type>OU</Type>
  <Route>XO</Route>
  <CPN>200</CPN>
  <CDPN>18201008888</CDPN>
  <TimeEnd>20170309161117</TimeEnd>
  <Duration>12</Duration>
  <TrunkNumber>02161205555</TrunkNumber>
  <Recording>20170309/200_18201008888_20170309_161105_8021_cg.wav</Rec
ording>
  <RecCodec>PCMU</RecCodec>
</Cdr>
```

◦ **OU call record of the menu which acts as calling party:** Menu calls the external phone 18201008888, the duration of the call is 4 seconds and the corresponding recording is that of the trunk 888.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="2520170309162028-0">
   <callid>32809</callid>
   <outer id="41" />
   <TimeStart>20170309162021</TimeStart>
   <Type>OU</Type>
   <Route>IP</Route>
   <CPN></CPN>
   <CDPN>18201008888</CDPN>
   <TimeEnd>20170309162028</TimeEnd>
   <Duration>4</Duration>
   <TrunkNumber>888</TrunkNumber>

<Recording>20170309/888_18201008888_20170309_162024_8029_cg.wav</Reco
rding>
   <RecCodec>PCMU</RecCodec>
</Cdr>
```

2.5.3 LO

- **When an extension calls another extension, 2 call records are generated, respectively as LO and IN (The IN call record can be filtered out)**

1. **LO call record of the extension 200 which acts as the calling party:** Extension 200 calls another extension 212, the duration of the call is 13 seconds and the corresponding recording is that of the extension 200.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="5720170309175055-0">
   <callid>32858</callid>
   <TimeStart>20170309175037</TimeStart>
   <Type>LO</Type>
   <Route>IC</Route>
   <CPN>200</CPN>
   <CDPN>212</CDPN>
```

```
  <TimeEnd>20170309175055</TimeEnd>
  <Duration>13</Duration>
  <TrunkNumber></TrunkNumber>
  <Recording>20170309/200_212_20170309_175042_805A_cg.wav</Recording>
  <RecCodec>PCMU</RecCodec>
</Cdr>
```

2. **LO call record of the extension 212 which acts as the called party:** Extension 200 calls another extension 212, the duration of the call is 4 seconds and the corresponding recording is that of the extension 212.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="2220170309161649-0">
  <callid>32806</callid>
  <TimeStart>20170309161641</TimeStart>
  <Type>IN</Type>
  <Route>IC</Route>
  <CPN>200</CPN>
  <CDPN>212</CDPN>
  <TimeEnd>20170309161649</TimeEnd>
  <Duration>4</Duration>
  <TrunkNumber></TrunkNumber>
  <Recording>20170309/200_212_20170309_161644_8026_cd.wav</Recording>
  <RecCodec>PCMU</RecCodec>
</Cdr>
```

2.5.4 FI/FW

1. When the call transfer number is an internal extension number, only the FI call record is generated.

- **FI call record of the extension which acts as the called party:** The external phone 02167103750 calls the extension 200, but the extension 200 transfers the call to extension 201, and the duration of the call is 4 seconds.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="5820821018163856-0">
  <callid>90</callid>
  <visitor id="89" />
  <TimeStart>20821018163827</TimeStart>
  <Type>FI</Type>
  <Route>IC</Route>
  <CPN>02167103750</CPN>
  <CDPN>200</CDPN>
  <TimeEnd>20821018163856</TimeEnd>
  <Duration>4</Duration>
  <TrunkNumber>02161208234</TrunkNumber>
</Cdr>
```

2. When the call transfer number is an external call or cell phone number, two single words are generated, respectively as FI and FW.

- **FI call record of the extension which acts as the called party:** The external phone 02167103750 calls the extension 200, but the extension 200 transfers the call to the cell phone 13012345678, and the duration of the call is 4 seconds.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="5820821018163856-0">
  <callid>90</callid>
  <visitor id="89" />
  <TimeStart>20821018163827</TimeStart>
  <Type>FI</Type>
  <Route>IC</Route>
  <CPN>02167103750</CPN>
  <CDPN>200</CDPN>
  <TimeEnd>20821018163856</TimeEnd>
  <Duration>4</Duration>
  <TrunkNumber>02161208234</TrunkNumber>
</Cdr>
```

- **FW call record of the extension which acts as the called party:** The external phone 02167103750 calls the extension 200, but the extension 200 transfers the call to the cell phone 13012345678, and the duration of the call is 4 seconds.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="5920821018163856-0">
  <callid>90</callid>
  <visitor id="89" />
  <TimeStart>20821018163827</TimeStart>
  <Type>FW</Type>
  <Route>XO</Route>
  <CPN>200</CPN>
  <CDPN>13012345678</CDPN>
  <TimeEnd>20821018163856</TimeEnd>
  <Duration>4</Duration>
  <TrunkNumber></TrunkNumber>
</Cdr>
```

2.5.5 CB

1. Two-way outbound call produces two CB call records.

- **CB call record of the external phone which acts as the calling party:** The external phone 13012345678 calls another external phone 02167103750, and the duration of the call is 9 seconds.

```
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="420170328135454-0">
  <MAC>00:0E:A9:3D:00:0C</MAC>
  <callid>20484</callid>
  <outer id="4" />
  <TimeStart>20170328135442</TimeStart>
  <Type>CB</Type>
  <Route>IP</Route>
  <CPN>13012345678</CPN>
  <CDPN>02167103750</CDPN>
  <TimeEnd>20170328135454</TimeEnd>
  <Duration>9</Duration>
  <TrunkNumber>50015</TrunkNumber>
<Recording>20170328/50015_02167103750_20170328_135445_5004_cg.wav</Record
ing>
  <RecCodec>PCMU</RecCodec>
</Cdr>
```

- **CB call record of the external phone which acts as the called party:** The external phone 13012345678 calls another external phone 02167103750, and the duration of the call is 6 seconds.

```xml
<?xml version="1.0" encoding="utf-8" ?>
<Cdr id="520170328135455-0">
   <MAC>00:0E:A9:3D:00:0C</MAC>
   <callid>20484</callid>
   <outer id="5" />
   <TimeStart>20170328135445</TimeStart>
   <Type>CB</Type>
   <Route>IP</Route>
   <CPN></CPN>
   <CDPN>02167103750</CDPN>
   <TimeEnd>20170328135455</TimeEnd>
   <Duration>6</Duration>
   <TrunkNumber>50016</TrunkNumber>
<Recording>20170328/50016_02167103750_20170328_135449_5005_cg.wav</Record
ing>
   <RecCodec>PCMU</RecCodec>
</Cdr>
```

3 Implementations

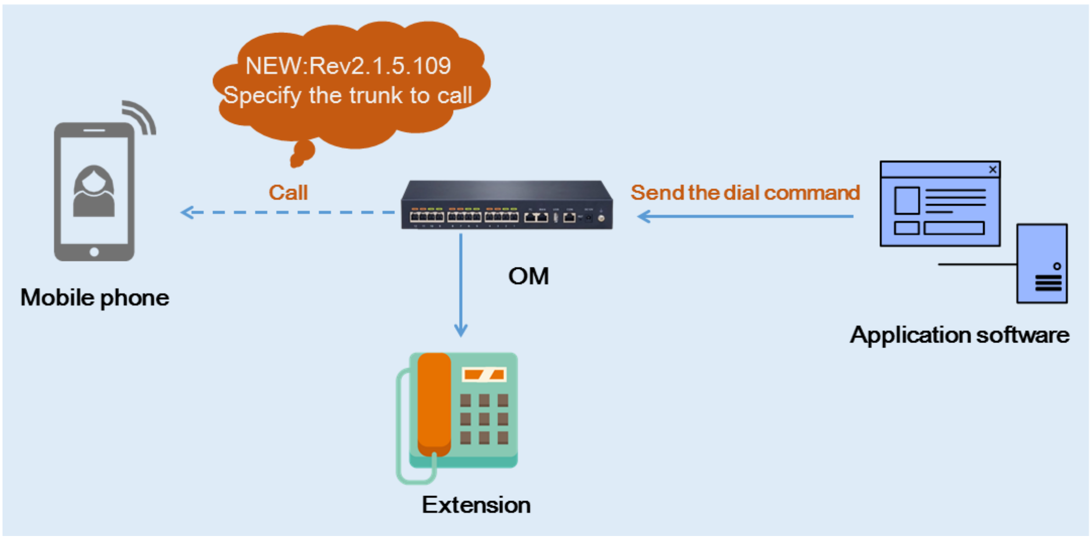
This chapter describes some of the common implementations. See below for more details.

3.1 Click-to-dial

Click-to-dial is a common functionality of call centers, CRM systems and other application softwares. As soon as the user clicks on a contact on the software or Web, the system automatically initiates a call to the contact, which make the call easier and more efficient.

Realization principle

The schematic diagram is as follows:



Note: Application server and client are developed by developers.

Steps description

When the user clicks on a contact on the application client or web, the client sends a dial request to the application server.

1. The application server determines whether the calling number belongs to an extension or external call. If it's an extension, then excute the command of extension

calling extension. If it’s an external telephone, then excute the command of extension calling external phone. Of course, the calling extension number (ext, id) is the user's local number at this point.

2. The application server sends the corresponding API command to OM through the HTTP protocol, and after the message is received, OM performs authentication first, and then executes the command after the authentication is passed.
3. OM executes the dialing command to call the calling party and the called party in a certain order, and when the two parties answer, the call is established.

Characteristics of click-to-dial

1. Support two call modes:

- High efficiency type——Call the called party first and call the calling party after received the call back signal.
- Good experience type ——Call the calling party first, and call the called party after the calling party offhooks.

2. Support the the automatic response function of SIP entity phone who acts as the calling party

Call command

Click-to-dial commands include "extension calling extension" and "extension calling external phone”.

- **API command of “extension calling extension”**

```
<?xml  version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
  <ext id="200"/>
  <ext id="208"/>
</Transfer>
```

- **API command of “extension calling external phone”**

```
<?xml  version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
  <ext id="208"/>
  <outer to="13012345678"/>
</Transfer>
```

3.2 Pop-up of an incoming/outbound call

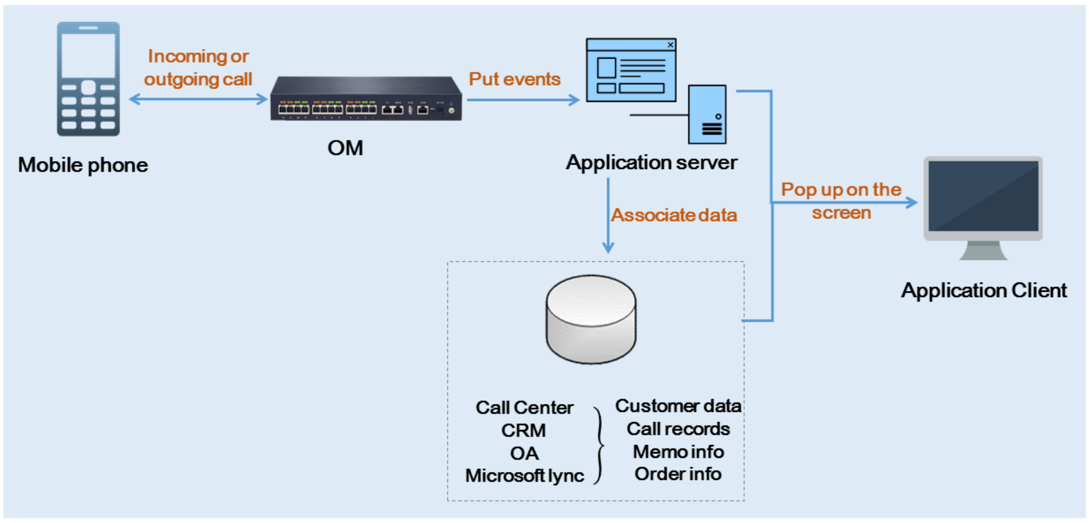
Definition: When an extension of the customer service receives a call from a customer or initiates a call to a customer, the computer automatically pops up with displaying the customer's details.

Objective: To improve the efficiency and quality of service.

Realization principle

1. In the process of completing a call, the call status will change continuously, such as ringing, connecting, hanging, etc. OM will push these event messages to the application server developed by the developer in real time.
2. After receiving the messages, the application server queries the corresponding customer data, history call records and other information of the database, and then controls the application terminal which the extension binds to perform the actions of pop-up screen.

Schematic diagram:



When the application server executes the pop-up screen:

There are 8 call status change events of the extension, respectively as RING, ALERT, ANSWER, ANSWERED, BYE, DIVERT, TRANSIENT and FAILED.

Events associated with the screen: RING, ALERT and TRANSIENT.

Pop-up screen

After receiving the RING event containing ext and visitor, execute the call screen.

Message sample:

Mobile phone 13012345678 calls extension 200, the extension 200 starts ringing:

```
<?xml version="1.0"   encoding="utf-8" ?>
<Event attribute="RING">
    <ext id="200" />
    <visitor id="22"   from="13012345678" to="02161208234"   callid="16408" />
</Event>
```

Outbound call screen

After receiving the ALERT and TRNASIENT events containing outer and ext, execute the pop-up screen.

- The TRANSIENT event is for analog trunk;
- The ALERT event is primarily for SIP trunk.

Message sample:

1. Extension 200 calls the external phone 13012345678 via the anolog trunk:

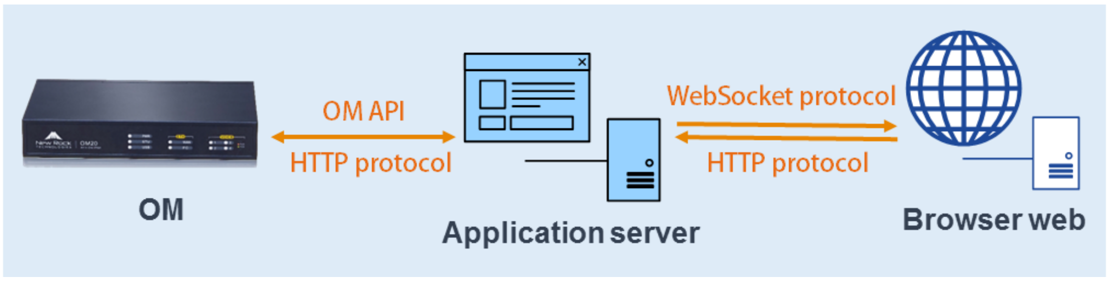
```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="TRANSIENT">
 <outer id="2" from="200" to="13012345678" trunk="306" callid="16386" />
 <ext id="200" />
</Event>
```

2. Extension 200 calls the external phone 13012345678, and OM checks call back information of the external call (Namely, external call begins ringing):

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ALERT">
 <outer id="2" from="200" to="13012345678" trunk="306" callid="16386" />
 <ext id="200" />
</Event>
```

How to implement the pop-up screen under the B/S framework:

After receiving the API event, the application server forwards the events to the browser to inform the screen to pop-upo. At this point, because the HTTP protocol does not support the server accessing to the browser actively, so it is necessary to adopt another protocol, the WebSocket protocol.



3.3 Multi-level Interactive Voice Responsevoice (Multi-level IVR)

Multi-level IVR is a common application of call centers.

The object of voice menu (menu) should be used to implement the multi-level IVR.

Configuration requirements

Open the switches of “Post-answer control for incoming” and “Status subscription” of the trunks.

Note: Post-answer control for incoming, that is, after the incoming call is transferred to OM, whether the call should be transferred to the IVR or artificial is controlled by the application server completely.

Related API

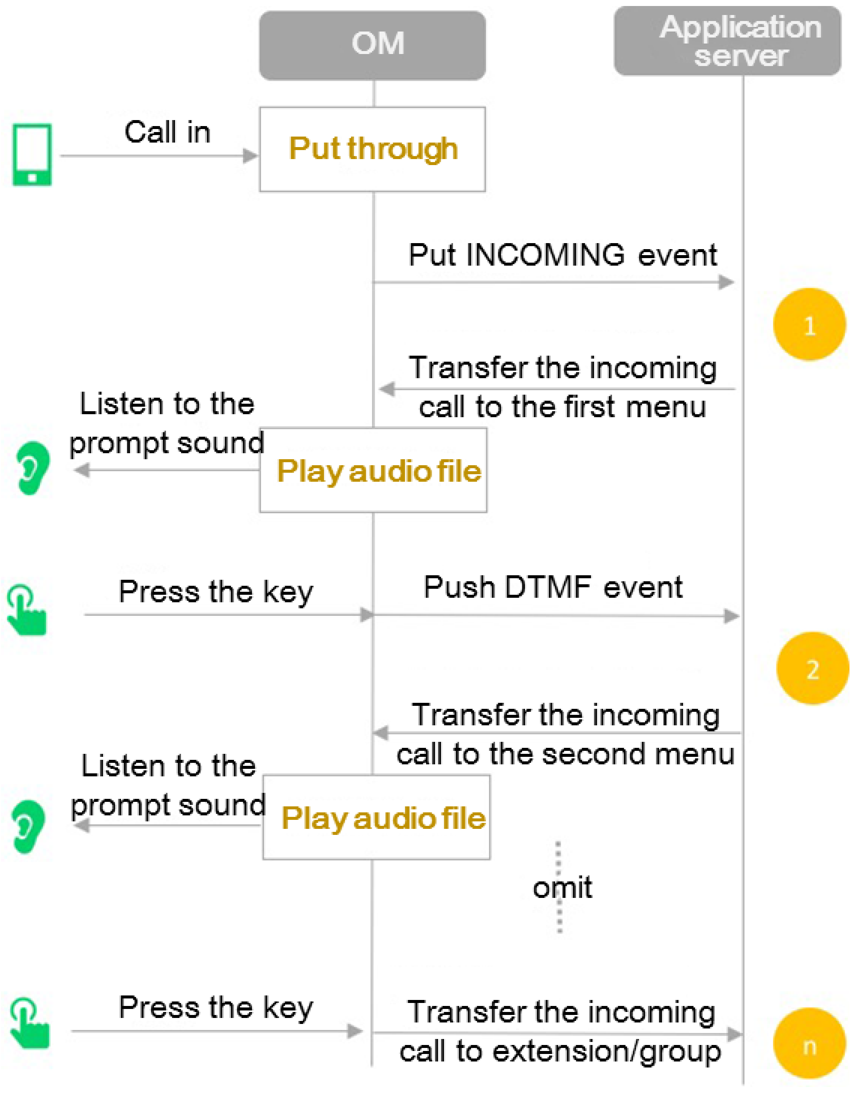
Events:

- **INCOMING event:** Turn on the switch of “Post-answer control for incoming”, when the incoming call calls the trunk, om will answer the call first, and push the INCOMING event, and then wait the application server to control the call;
- **DTMF event:** After the menu voice file is played, and when the button pressed by user meets the condition of menu key information reporting, the OM pushes this event to the application server;
- **EndOfAnn event:** When the call is transferred to menu, and after the menu voice file is played, OM pushes this event to the application server.

Request:

- **Assign request:** Configure the voice menu (menu) for later use. Configuration items include the voice files, playing times, key detection length, and the end character of the input;
- Connect request: Used to jump and transfer between menus and manual, such as transferring incoming call to menu, transferring incoming call to extension.

Simple implementation chart of IVR:


OM
Application server
Call in
Put through
Put INCOMING event
1
Transfer the incoming call to the first menu
Listen to the prompt sound
Play audio file
Press the key
Push DTMF event
2
Transfer the incoming call to the second menu
Listen to the prompt sound
Play audio file
omit
Press the key
Transfer the incoming call to extension/group
n

Exception handling

If the user does not press button as the IVR, it is necessary that the application server handles the exceptions.

Abnormal phenomenon:

- **Judging input timeout:** The value of info in the DTMF event ends with the letter T, which indicates the the button is entered overtime;
- **Judging input error:** The value of info in the DTMF event is not within the valid value defined by this menu, which indicates that the entered button is error.

Example:

1. DTMF event of the first input timeout:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="4" from="13012345678" to="02161208234" callid="36868">
        <info>T</info>
        <menu id="1" />
    </visitor>
</Event>
```

2. DTMF event of timeout with inputing half:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="4" from="13012345678" to="02161208234" callid="36868">
        <info>1234T</info>
        <menu id="1" />
    </visitor>
</Event>
```

Exception handling:

When these circumstances occur, play the error correction prompt to users generally, after that transfer the call back to the original menu.

However, there is a simpler way to turn back to the original menu and play an error correction prompt. That is:

Emporarily replace voice files:

Format such as:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
```

```
    <visitor id="41" />
    <menu id="10"/>
    <voicefile>input_wrong＋input_id_card_no</voicefile>
</Transfer>
```

Temporarily replace speech file descriptions:

- If the specified temporary new voicefile calls menu, OM no longer play menu configuration of the original voicefile to the party, and play the temporary designated voicefile, and only played once, after the voicefile is played, OM pushes EndOfAnn event report;
- The reporting rules of DTMF still follow the menu configuration (i.e., the two parameters of infolength and exit are still valid as menu configuration);
- The temporary designated voicefile will only be valid for this transfer.

Example：

For example, the configuration of menu 10 is:

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Assign">
    <menu id="10">
        <voicefile>input_id_card_no</voicefile>
        <repeat>1</repeat>
        <infolength>20</infolength>
        <exit>#</exit>
    </menu>
</Control>
```

Among them, the contents of the input_id_card_no voice file: “Please enter your ID number, end by the key of #”.

1. When the incoming call is transferred to the menu, user hears the tone "Please enter your ID number, end by the key of #", then the user inputs "123#", the application server receives a DTMF event:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="41" from="13012345678" to="02161208234" callid="36868">
        <info>123#</info>
        <menu id="1" />
    </visitor>
</Event>
```

2. When the application server determines that the user input data is error, it will still transfer the call to menu 10, and assign two voice files like Input_wrong + input_id_card_no, of which the content of the voice file input_wrong is "Your input is incorrect", and the symbol “+” is used to connect multiple files).

So, the incoming call is still in menu 10, but the prompt tone becomes “You have entered wrong, please enter your ID number, aslo end by the key of #”.

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="41" />
    <menu id="10"/>
    <voicefile>input_wrong＋input_id_card_no</voicefile>
</Transfer>
```

3. Then, the user entered the correct ID number “410410199001010101#”, and the application server receives:

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="41" from="13012345678" to="02161208234" callid="36868">
        <info>410410199001010101#</info>
        <menu id="1" />
    </visitor>
</Event>
```

Thus, an exception handling is completed.

3.4 Blacklist for incalling calls

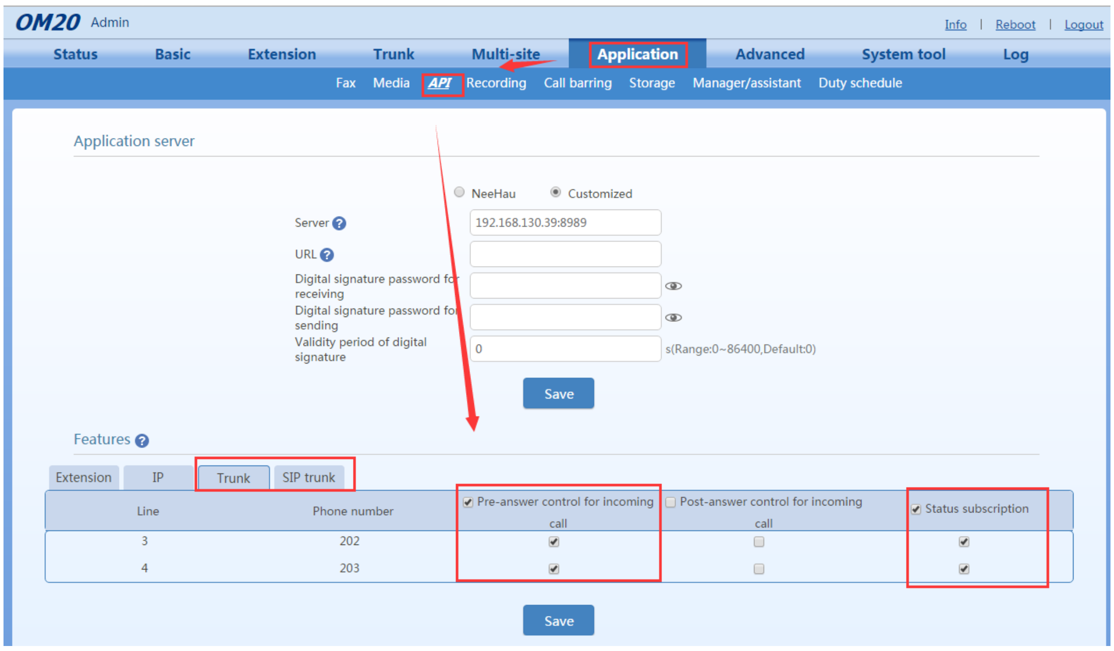
To implement call blacklist, it’s necessary to open the switches of “Pre-answer control for incoming call” and “Status subscription”.

Explanation of the switch of “Pre-answer control for incoming”

- If the switch is turned on, when a caller calls corresponding trunk, OM will push INVITE event to the application server, and wait the application to control the call. If the call is expected to accept, the application server will execute the Accept API request. If it is expected to deny, the application server will execute the Clear API request. And if the application server hasn't excute some control operation yet, the OM will automatically receive the call after waiting for a timeout (Five seconds by default).

- If the switch is closed, OM automatically accepts the call by default.

Configuration as shown below:


OM20 Admin
Info | Reboot | Logout
Status Basic Extension Trunk Multi-site Application Advanced System tool Log
Fax Media API Recording Call barring Storage Manager/assistant Duty schedule

Application server

( ) NeeHau (● Customized)

Server	192.168.130.39:8989
URL	
Digital signature password for receiving	
Digital signature password for sending	
Validity period of digital signature	0 s(Range:0~86400,Default:0)

Save

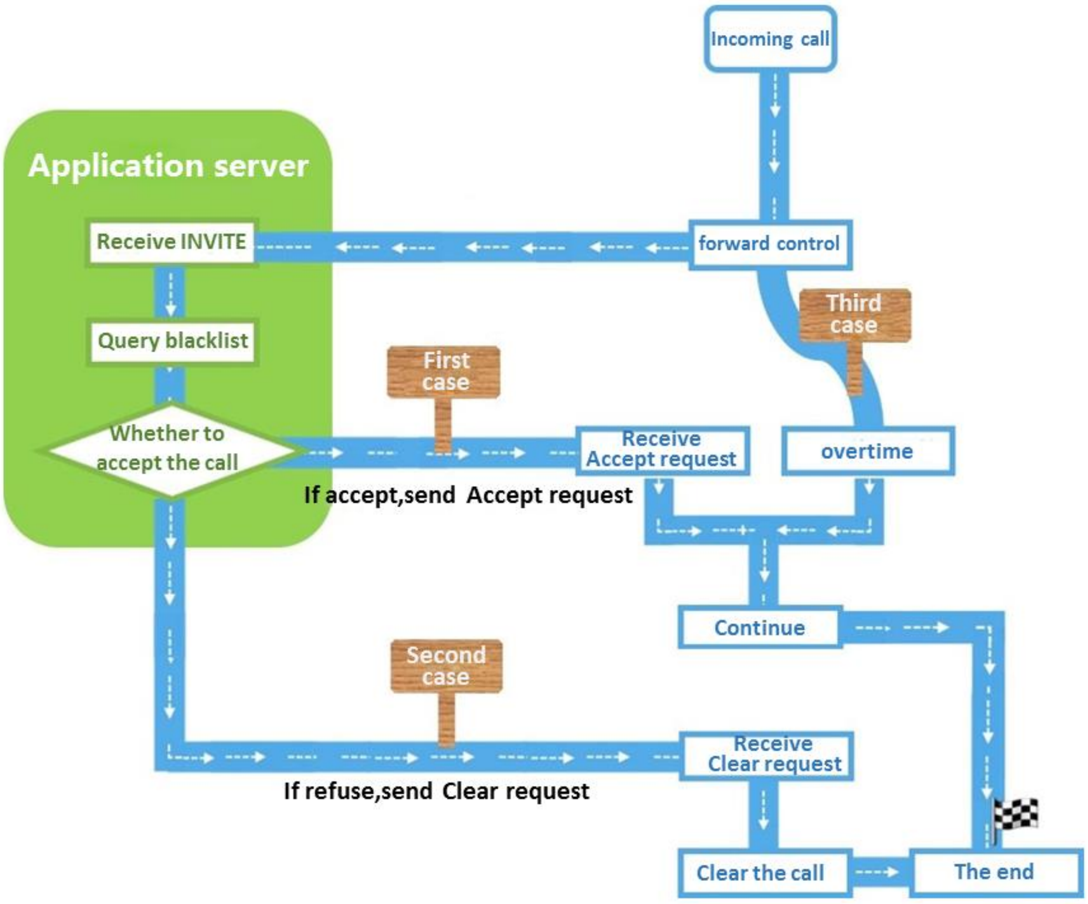
Features

Extension IP Trunk SIP trunk

Line	Phone number	✓ Pre-answer control for incoming call	☐ Post-answer control for incoming call	✓ Status subscription
3	202	✓	☐	✓
4	203	✓	☐	✓

Save

Realization process of calling blacklist:



The API uesed

- INVITE event (Incoming call request event);
- Accept request (Accept the incoming call);
- Clear request (Reject the incoming call).

3.5 Outbound call via specified trunk

There are two ways to call the external phone via specified trunk, respectively through adding prefix or specifying the trunk id.

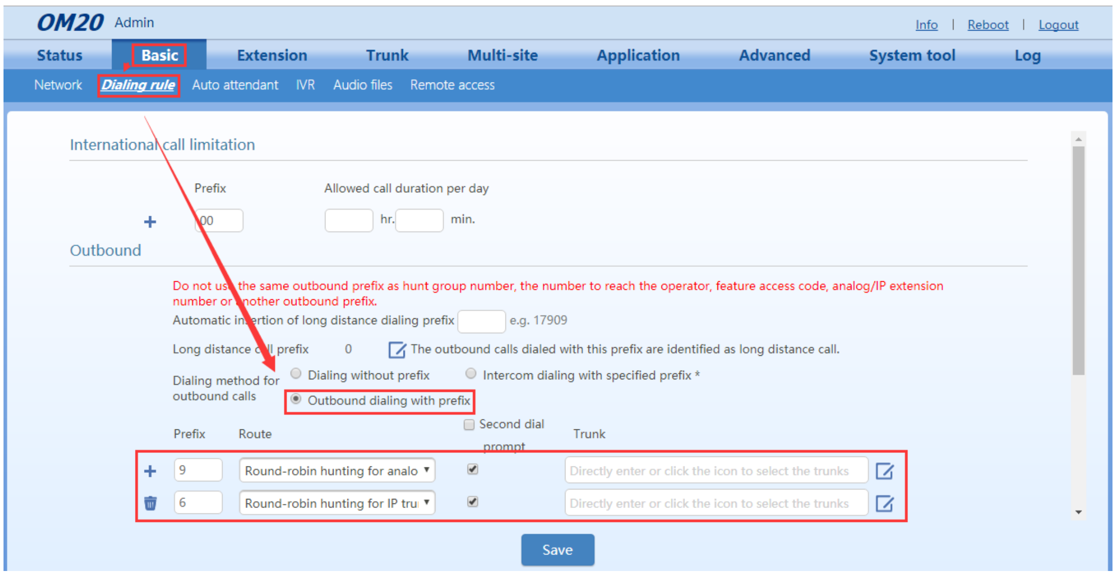
1. Outbound call via adding prefix

Adding prefix before the called number, any idle IP or anolog trunk will be selected according to the dial rule when you execute an outbound call.

Call conditions

OM needs to configure prefix outbound rules:

Login OM, enter **basic > Dialing rule > Outbound**, choose "Outbound dialing with prefix", and then configure prefix and line selection mode. As shown in the following figure:



International call limitation

Prefix	Allowed call duration per day
00	hr. min.

Outbound

Do not use the same outbound prefix as hunt group number, the number to reach the operator, feature access code, analog/IP extension number or another outbound prefix.

Automatic insertion of long distance dialing prefix e.g. 17909

Long distance call prefix 0 The outbound calls dialed with this prefix are identified as long distance call.

Dialing method for outbound calls: Dialing without prefix Intercom dialing with specified prefix * [✓ Outbound dialing with prefix]

Prefix	Route	Second dial prompt	Trunk
9	Round-robin hunting for analo	✓	Directly enter or click the icon to select the trunks
6	Round-robin hunting for IP tru	✓	Directly enter or click the icon to select the trunks

Command format

For example, the outbound rules of OM is that the prefix is 9, all the outbound calls go out via SIP trunk, and the called number is 13012345678, thus the command format is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
        <ext id="208"/>
        <outer to="9,13012345678"/>
</Transfer>
```

2. Outbound call via specified trunk id

Version requirements: Rev2.1.5.109 and above.

Command format

Increase the <trunk id> node in outbound API interface command, and the call wil be from a fixed trunk.

For example, specify the trunk 888, and the command format is:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
        <ext id="208"/>
```

```
    <trunk id="888" />
    <outer to="13012345678"/>
</Transfer>
```

Parameter description:

Parameter name	Type	Description
<ext id>	string	Extension number
<outer to>	string	Outbound called number
[trunk id]	string	The trunk number, can be either IP or analog trunk.

Supported scenarios of specified outbound trunk include: **extention calling external phone, transferring incoming call to external phone, transferring outbound call to external phone, menu calling external phone.**

Special case

Two-way outbound call is special, it needs with the help of the display field to specify fixed trunk. Format is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
   <outer to="13012345678" display="50018" />
   <outer to="02167103750" display="50019" />
</Transfer>
```

Parameter description:

Parameter name	Type	Description
<outer to>	string	External phone number. The first one is the calling number and the second one is the called number.
<display>	string	Specified trunk number, may be IP or anolog trunk. Related parameters include API_DISPLAY and API_CPN.

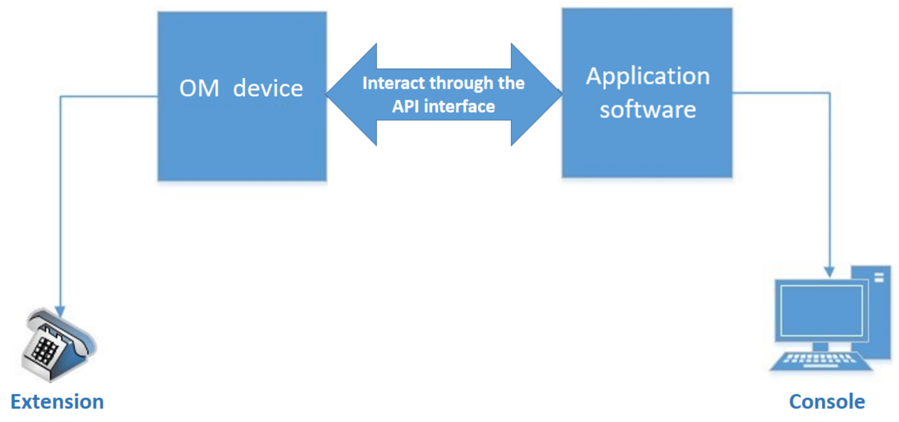
3.6 Hotel wake up solution

Scene description

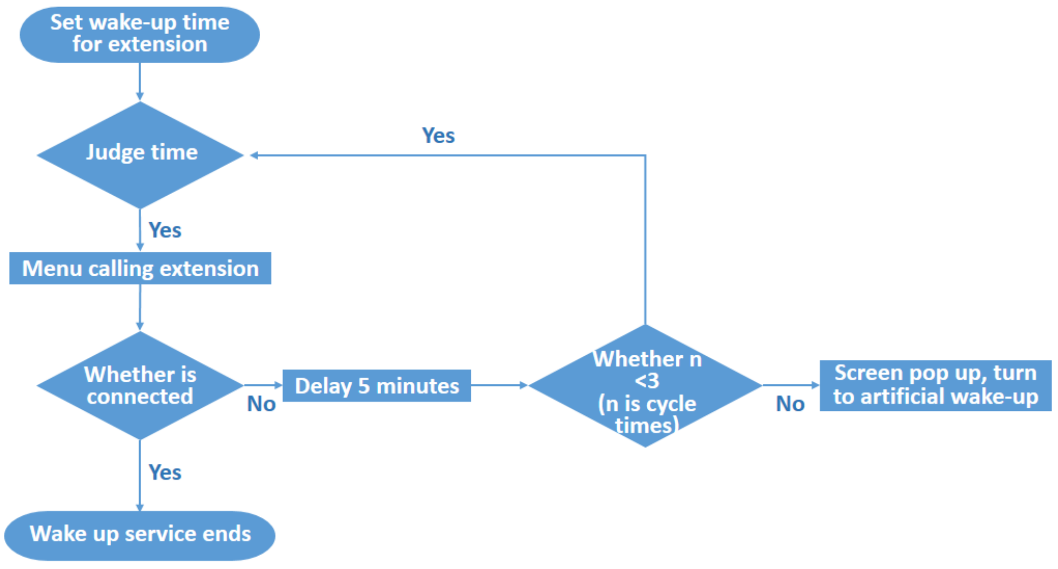
1. Apply for wake-up service at the hotel counter, and provide room number and wake-up time.
2. The hotel staff inputs the above information into the “hotel wake-up system”.
3. When the wake-up time arrives, the “hotel wake-up system” controls the IP-PBX to initiate a call to the extension in guest room automatically.
4. When the telephone in guest room rings, and the guest answers the call, the automatic wake-up call is complecated.
5. If there is still no person answer it after two times executing wake-up call automatically, the hotel wake-up system pops up, reminding the staff to perform artificial wake-up service.

Implementation scheme

The schematic diagram is as follows:



Implementation process:


Set wake-up time for extension
Judge time
Yes
Yes
Menu calling extension
Whether is connected
No
Delay 5 minutes
Whether n <3 (n is cycle times)
No
Screen pop up, turn to artificial wake-up
Yes
Wake up service ends

Process description:

1. Set up wake-up time and the room extension number which needs to be wake up via PC console.

2. When the wake-up call arrives, the application software sends XML text messages to OM through the API interface: **menu calling extension**, and the extension rings.

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <menu id="1"/>
    <ext id="212"/>
</Transfer>
```

3. If the guest answers the phone and the application server receives the ANSWER event shown as below, which means the guest is woken up. Otherwise, the application server will sends the command of menu calling extension again after 5 minutes.

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="ANSWER">
    <ext id="212" />
    <menu id="1"/>
</Event>
```

4. If there is still no person answers the call after sending two times, the console executes pop-up screen to remind the staff to wake up the guest artificially.

3.7 Implementation scheme of playing voice file after connected

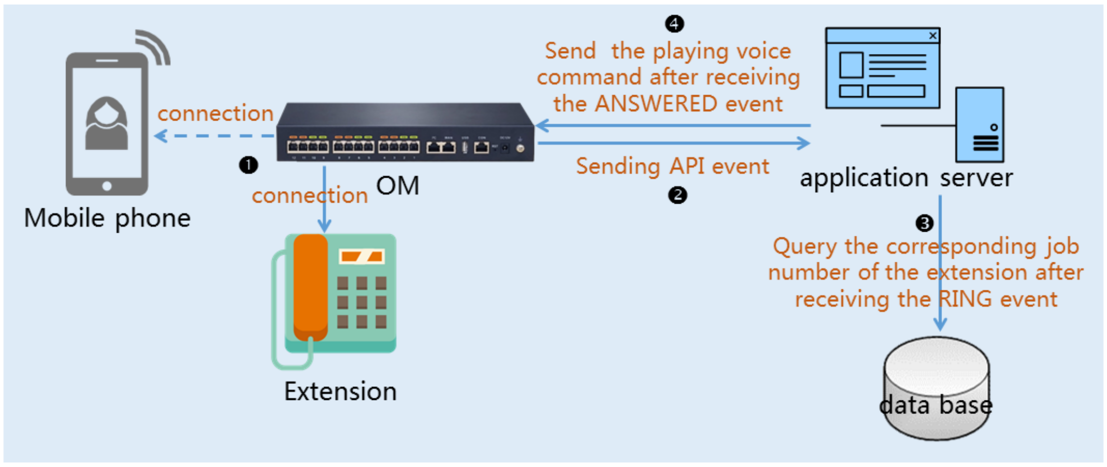
Application scenario

After the incoming call and extension are connected, play voice file to the incoming call, such as “2202 operator for your service.”

Limitations

Version requirements: Rev 2.1.5.117 and above.

Implementation diagram


❹
Send the playing voice command after receiving the ANSWERED event
connection
Mobile phone
❶
connection
OM
Sending API event
❷
application server
❸
Query the corresponding job number of the extension after receiving the RING event
Extension
data base

Implementation process

Step 1: Record the voice file, such as service_1: "Number operator for you", AND upload it to OM;

Step two: Execute the API of incoming call forwarding extension, specify the temporary voice files as two mute files, command is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor id="14"/>
    <ext id="200"/>
    <voicefile>silence+silence</voicefile>
</Transfer>
```

Step three: after receiving an answering event (ANSWERED) from the trunk, execute the interface of inserting voice file.

API for playing voice file to incoming call:

```
<?xml version="1.0" encoding="udf-8" ?>
<Transfer attribute="Connect">
    <visitor id="15" />
    <voicefile>2202+service_1</voicefile>
</Transfer>
```

Description:

- "2202" is the corresponding job number of the extension;
- When the incoming party listens to the voice, the extension mutes. And after the voice playback, they enter into the call automatically;
- When the incoming call listens to the voice, if you want the extension to listen to a hint tone, you can also insert the voice into the extension at the same time.

API for playing voice file to an extension:

```
<?xml version="1.0" encoding="udf-8" ?>
<Transfer attribute="Connect">
    <ext id="200" />
    <voicefile>voice file name</voicefile>
</Transfer>
```

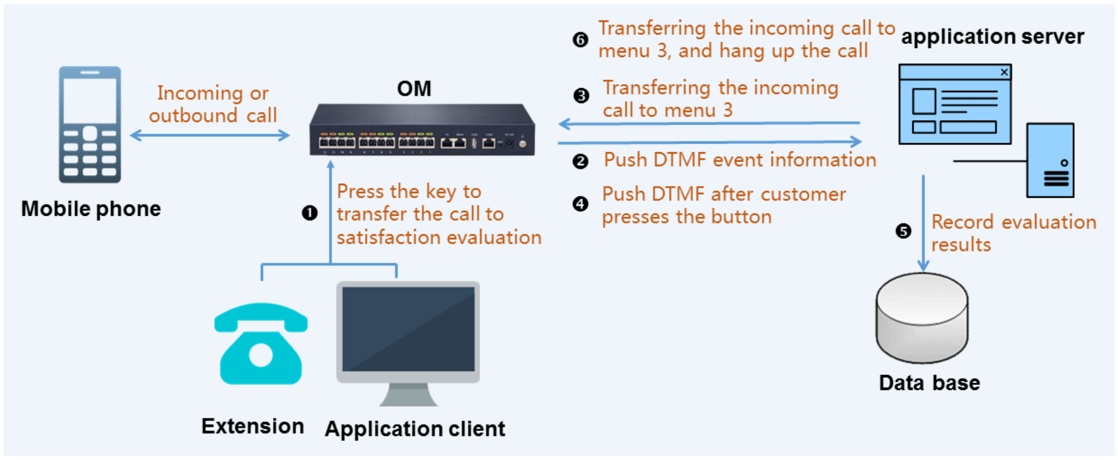
Note: Click here to find out the detailed interface of playing voice file.

3.8 Implementation scheme of satisfaction evaluation

Application scenario

After the call is finished between the incoming/outgoing call and customer service, it will be transferred to a satisfaction evaluation system. If the customer gives an evaluation in specified time, the system will record the evaluation results, otherwise, the call will be hung up by the application server.

Implementation diagram


Mobile phone
Incoming or outbound call
OM
❶ Press the key to transfer the call to satisfaction evaluation
Extension
Application client
❻ Transferring the incoming call to menu 3, and hang up the call
❸ Transferring the incoming call to menu 3
❷ Push DTMF event information
❹ Push DTMF after customer presses the button
application server
❺ Record evaluation results
Data base

Environmental requirements

Version requirements: Rev 117 and above;

Permission configuration: Configure extension (customer service phone), whose API parameter value is oxF.

Implementation process

1. Configure voice files, which is used for the voice prompt of satisfaction evaluation.

    1) **Voice file 1:** Very satisfied, please press 1, basically satisfied, please press 2, not satisfied, please press 3.

```
<?xml version="1.0" encoding="utf-8" ?>
 <Control attribute="Assign">
     <menu id="1">
         <voicefile>evaluate</voicefile>
         <repeat>3</repeat>
         <infolength>1</infolength>
         <exit>#</exit>
     </menu>
```

```
    </Control>
```

2) **Voice file 2:** You have timed out and the call is about to be hung up.

```
   <?xml version="1.0" encoding="utf-8" ?>
    <Control attribute="Assign">
          <menu id="2">
                <voicefile>timeout</voicefile>
                <repeat>1</repeat>
                <infolength>1</infolength>
                <exit>#</exit>
          </menu>
    </Control>
```

3) **Voice file 3:** Thank you for your comments.

```
     <?xml version="1.0" encoding="utf-8" ?>
      <Control attribute="Assign">
            <menu id="3">
                  <voicefile>thanks</voicefile>
                  <repeat>1</repeat>
                  <exit>#</exit>
            </menu>
      </Control>
```

2. During the call, the customer service presses button through the phone, meanwhile OM reports DTMF event to the application server. If the customer service has application system, you can skip this step, and do the key transfer satisfaction evaluation in the application service system.

   1) DTMF event of outbound call

```
      <?xml version="1.0" encoding="utf-8" ?>
      <Event attribute="DTMF">
            <ext id="300" />
            <info>5</info>
```

```
        <outer id="1" from="300" to="788" trunk="309" callid="32769"    />
    </Event>
```

2) DTMF event of incoming call

```
   <?xml version="1.0" encoding="utf-8" ?>
    <Event attribute="DTMF">
        <ext id="300" />
        <info>7</info>
        <visitor id="1" from="788" to="309" callid="32772"    />
    </Event>
```

3. According to the key events or satisfaction evaluation request, the application server transfers the call to menu 1 to request being given evaluation.

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
  <visitor/outer id="1" />
  <menu id="1"/>
</Transfer>
```

4. Waiting to receive the customer's evaluation results. If no customer's key information is received (DTMF event of menu 1)

   1) If you have not receive the DTMF event after receiving the EndOfAnn event by delaying 5 seconds, you can excute step 3 again. And if you still haven't receive the customer's key information after repeating two times, to excute the API of transferring the incoming call to menu 2.

```
   <?xml version="1.0" encoding="utf-8" ?>
    <Transfer attribute="Connect">
        <visitor/outer id="1" />
        <menu id="2"/>
    </Transfer>
```

   2) When receiving the EndOfAnn event in menu 2, release the call.

```
<?xml version="1.0" encoding="utf-8" ?>
<Control attribute="Clear">
    <visitor/outer id="1"/>
</Control>
```

5. Waiting for the customer’s button information and receiving it (The DTMF event of menu 1)

   1) Record the result of satisfaction evaluation results, that is, the value of info.

```
<?xml version="1.0" encoding="utf-8" ?>
<Event attribute="DTMF">
    <visitor id="3" from="788" to="309" callid="32771">
        <info>1</info>
        <menu id="1" />
    </visitor>
</Event>
```

   2) After receiving the DTMF event, transferring the call to menu 3, prompting thanks to the evaluation.

```
<?xml version="1.0" encoding="utf-8" ?>
<Transfer attribute="Connect">
    <visitor/outer id="1" />
    <menu id="3"/>
</Transfer>
```

   3) When receive the EndOfAnn event of the voice menu 3, release the call.

```
<?xml version="1.0" encoding="utf-8" ?>
  <Control attribute="Clear">
      <visitor/outer id="1"/>
  </Control>
```

4 Appendix

This chapter describes the OM system parameters and their query and configuration methods.

4.1 System parameter list

(**Tips:** shortcut links are returned with Alt + ←)

Parameter name	**id**	**parameter values**	**Description**
API_MAC	349	yes，no. The default value is no.	Yes: When OM sends API messages, it brings MAC address of OM, in the form of xxxxxx No: No MAC will be carried.
API_RESPONSE_TO	526	1-120 seconds. The default value is 5 seconds.	After OM pushes INVITE or INCOMING event to the application server, if the application server does not respond in this time, OM will automatically process the call.
CONTROL_TYPE	554	1，0 The default value is 0.	1: OM pushes API report using TCP long connection. 0: OM pushes API report using TCP short connection.

Parameter name	**id**	**parameter values**	**Description**
FT_API_1	599	0，7，17，27，37 The default value is 0.	API switch parameter 0: shut down 7: status subscription 17: Pre-answer control for incoming + Status subscription 27: Post-answer control for incoming + Status subscription 37: Pre-answer control for incoming + Post-answer control for incoming + Status subscription **Note:** this parameter is per-line parameter. Click here to see the query and configuration method.
CB_TYPE	615	0，1 The default value is 0.	0: the first road of two-way outbound call calls out from IP trunk. 1: the first road of two-way outbound call calls out from analog trunk.

Parameter name	**id**	**parameter values**	**Description**
API_DISPLAY	624	yes，no The default value is yes.	Yes: The display number will be as the calling party No: The IP trunk number is used to call outer.
CONFIG_CHANGE	883	0，2 The default value is 0.	0: This event won’t be triggered when modify parameters by OM web; 2: This event will be triggered when modify parameters by OM web.
API_CALLING	978	yes，no The default value is no.	Yes: Click to dial, call the calling party first, and then call the called party after the calling party offhook. No: Click to dial, call the called party first, and then call the calling party after the called party ring back.

Parameter name	**id**	**parameter values**	**Description**
FT_ALERT_INFO_X	986	on，off The default value is off.	Auto answer parameter (X for line number) On: turn on auto answer Off: turn off the auto answer When opening INVITE carrying the Alert-Info field call IP extension, if the terminal supports automatically answered.
API_CPN	991	yes，no The default value is no.	Yes: API controls CPN, replacing the calling number; No: API displays API_DISPLAY.
API_OUTBOUND_TRANSFER	1070	0，1 The default value is 0.	0: Two external calls are not allowed to be transferred via API; 1: Allows two external calls to be transferred via API;

Parameter name	id	parameter values	Description
API_TIMEOUT	1088	0-86400 seconds The default value is 0.	Timeout time parameter of digital signature authentication 0: Permanent effect 1-86400: The authentication parameter is valid in a limited amount of time.
API_PASSWORD	1089	1-16 bit number or letter. The default value is null.	Digital signature authentication password Note: OM selects the IP authentication method at priority, and then selects digital authentication If IP authentication is not satisfied.
API_METHOD	1125	0，1 The default value is 0.	Parameter of API HTTP message push mode 0: use HTTP GET mode 1: use HTTP POST mode.

4.2 Query and configuration method of system parameters

(**Tips:** shortcut links are returned with Alt + ←)

OM API has some system parameters that are not open to the web page and the API interface, so they need to be modified through URL.

Query method

Open your browser and enter the URL path in the address bar:

- **For general parameters, URL format is as:**

http://OM IP/xml?method=gw.config.get&id=XX&pass=admin (Management password)

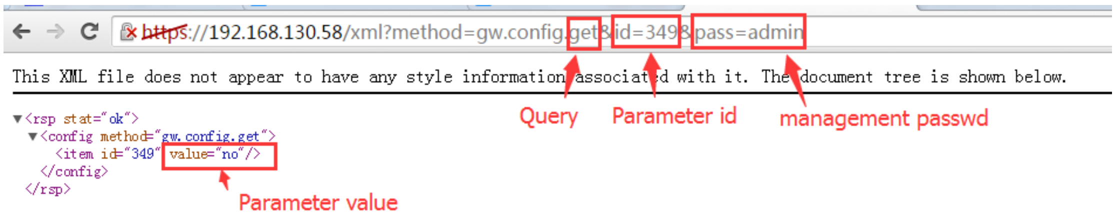
- **For pre-line parameters, URL format is as:**

http://OM IP/xml?method=gw.config.get&line_id=X&id=XX&pass=admin (Management password)

Note:

- line_id is the line number;
- System parameters include general parameters and pre-line parameters. For the information of parameters, you can click here to check.

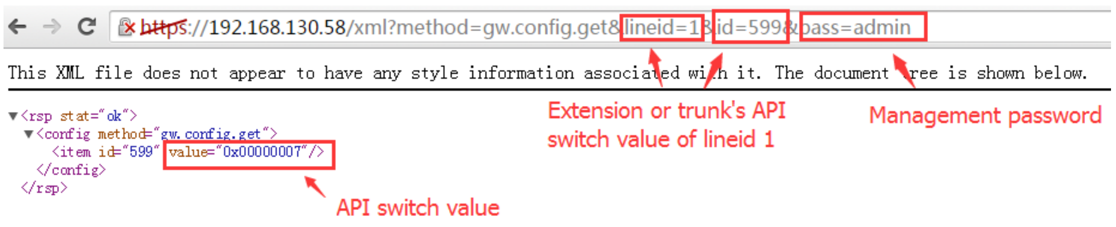
The example diagrams are shown below respectively:



~~https~~://192.168.130.58/xml?method=gw.config.get&id=349&pass=admin

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<rsp stat="ok">
  ▼<config method="gw.config.get">
      <item id="349" value="no"/>
    </config>
  </rsp>
```



~~https~~://192.168.130.58/xml?method=gw.config.get&lineid=1&id=599&pass=admin

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<rsp stat="ok">
  ▼<config method="gw.config.get">
      <item id="599" value="0x00000007"/>
    </config>
  </rsp>
```

Note: If you get the response of item value=”not defined”, possible reason is as: 1) The parameter name is error 2) The software version of OM is low.

Configuration method

Enter the URL path in the browser address bar:

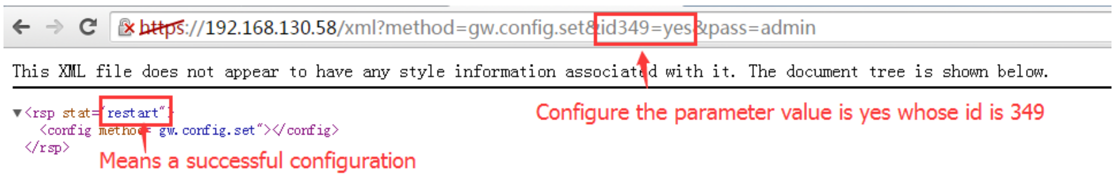
- **For general parameters, URL format is as:**

http://OM IP/xml?method=gw.config.get&idXX=XXX&pass=admin

- **For pre-line parameters, URL format is as:**

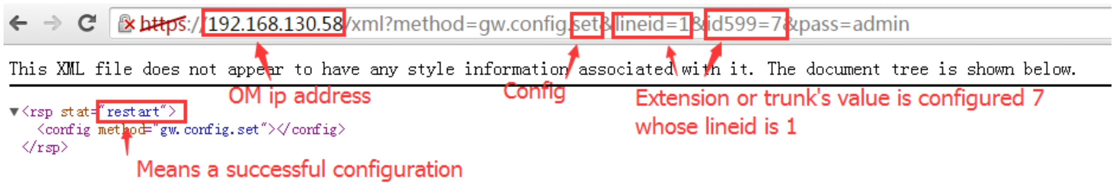
http://OM IP/xml?method=gw.config.get&line_id=X&idXX=XXX&pass=admin

The example diagrams are shown below respectively:



```
~~https~~://192.168.130.58/xml?method=gw.config.set&id349=yes&pass=admin

This XML file does not appear to have any style information associated with it. The document tree is shown below.

▼<rsp stat="restart">
   <config method="gw.config.set"></config>
 </rsp>
```

Configure the parameter value is yes whose id is 349
Means a successful configuration



```
~~https~~://192.168.130.58/xml?method=gw.config.set&lineid=1&id599=7&pass=admin

This XML file does not appear to have any style information associated with it. The document tree is shown below.

▼<rsp stat="restart">
   <config method="gw.config.set"></config>
 </rsp>
```

OM ip address
Config
Extension or trunk's value is configured 7 whose lineid is 1
Means a successful configuration

Response result

- If the response is rsp stat="restart", indicating that the configuration is successful;
- If the response is rsp stat="fail", indicating that the configuration is failure.