Component APIs

Component API Overview

In addition to the primary phone XSI API, the following two additional component APIs are available:

- Application Management API
- RTP Streaming API

Supported Phone Models

The following table lists the Cisco Unified IP Phone models that support the component APIs

Table 1: Phone Models that Support the Component APIs

Phone model	Supported, not supported	Firmware supported (see note 1)
Cisco Desk Phone 9800 Series		
9841	Supported	PhoneOS 3.0(1) or later
9851	Supported	PhoneOS 3.0(1) or later
9861	Supported	PhoneOS 3.0(1) or later
Cisco IP Phone 8800 Series		
8811	Supported	10.2(2) or later
8841	Supported	10.2(1) or later
8845	Supported	10.3(2) or later

Phone model	Supported, not supported	Firmware supported (see note 1)
8851	Supported	10.2(1) or later
8851NR	Supported	10.3(1) or later
8861	Supported	10.2(1) or later
8865	Supported	10.3(2) or later
8865NR	Supported	11.7(1) or later
Cisco IP Phone 8800 Series Multiplatform Phones	Supported	11.0(0) or later
Cisco Video Phone 8875 and 8875NR	Supported	PhoneOS 2.1 and later
Cisco IP Conference Phones		
7832	Not supported	—
8831	Not supported	—
8832	Supported	12.0(1) or later
Cisco Wireless IP Phone 8820 Series		
8821	Not supported	—
Cisco IP Phone 7800 Series		
7811	Not supported	—
7821	Not supported	—
7841	Not supported	—
7861	Not supported	—
Cisco IP Phone 7800 Series Multiplatform Phones	Not supported	—
Cisco IP Phone 6800 Series		
Cisco IP Phone 6800 Series with Multiplatform Firmware	Not supported	—

Note Cisco recommends the use of latest firmware. The firmware can be downloaded from the following location (requires login or service contract):

http://software.cisco.com/download/navigator.html?i=!mmd

Application Management API

To address the limited application management, the Application Management API provides a smoother handoff between the call mode and the application mode. The Application API consists of two primary components:

- Application URI
- Application Event Handlers

Note Support for the Application Management API requires an updated XML Parser.

The Multiplatform phones do not support the Application Management API.

Related Topics

Application Event Handlers
Application

RTP Streaming API

This XML-based RTP Streaming API allows applications to initiate and observe RTP audio streams. This API extends capabilities beyond the legacy RTP streaming URIs by providing support for stream start and stop event listeners and the ability to specify other extended stream attributes, such as codec type.

Note Support for the RTP Streaming API requires an updated XML Parser.

The Multiplatform phones doe not support the RTP Streaming API.

The event handlers typically use the standard Notification framework, but they can also invoke most other URIs, with the exception of HTTP URLs.

Interaction Rules with Legacy RTP URI Streams

The RTP Streaming API allows a full-duplex stream (mode=sendReceive) to be set up as a single stream request, which simplifies the usage of the API. However, in some cases, this API creates some interoperability issues with the legacy RTP URIs because the legacy RTP URIs send and receive streams separately. The interaction rules between legacy RTP URI streams and the new RTP Streaming API are:

- If an RTP Stop URI is invoked, and an RTP Streaming API stream is currently streaming in that same direction, then the entire RTP Streaming API stream is stopped.

  For example, if a full-duplex stream is set up through the RTP Streaming API (mode=sendReceive) and then an RTPTx:Stop URI is invoked, the stream will be stopped in both the send and receive directions (and the onStopped event handler will be called, if present).

- If the stopMedia request (from the RTP Streaming API) does not specify a stream ID, then the request will stop all services RTP streams, in any direction (send or receive) and of any type (multicast and

unicast). This allows applications using the RTP Streaming API to stop media streams which may have been started by the legacy RTP URIs or by other applications for which a stream ID is not known.

Error Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
  <xs:element name="errorResponse">
    <xs:complexType>
      <xs:all>
        <xs:element name="type">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="InvalidURL"/>
              <xs:enumeration value="InvalidResource"/>
              <xs:enumeration value="InvalidResourceID"/>
              <xs:enumeration value="UnavailableResource"/>
              <xs:enumeration value="InvalidXML"/>
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
        <xs:element name="data" nillable="true">
          <xs:simpleType>
            <xs:restriction base="xs:string"/>
          </xs:simpleType>
        </xs:element>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

RTP Streaming API Examples

The following examples show how to work with the RTP Streaming API.

Start Media Example

- Request

```
HTTP POST /CGI/Execute
<startMedia>
  <mediaStream
      onStopped=”Notify:http:server:80:path/page”
      receiveVolume=”50”>
        <type>audio</type>
        <codec>G.729</codec>
        <mode>sendReceive</mode>
        <address>239.1.2.3</address>
        <port>20480</port>
  </mediaStream>
</startMedia>
```

- Response

```
HTTP200 OK
<mediaStream id=”abc123”/>
```

Stop Media Example

- Request

```
HTTP POST CGI/Execute
<stopMedia>
  <mediaStream id=”abc123”/>
</stopMedia>
```

- Response

```
HTTP 200 OK
```

If the user terminates the media stream by placing the active audio path on-hook, the following notification is sent:

```
HTTP POST /server/path/page
DATA=<notifyMediaEvent type=”stopped” origin=”user”>
    <mediaStream id=”abc123”/>
</notifyMediaEvent>
```

Errors and Responses

The following table describes error conditions and responses for the RTP Streaming API.

Table 2: RTP Streaming API Error Conditions and Responses

Condition	Applicable method	HTTP result code	Type	Data
Authorization failed	all	401 (Authorization Failed)	N/A	N/A
Request object does not comply with the API’s XML schema	all	400 (BadRequest)	InvalidXML	<parser error description>
Media cannot be started because no DSP resources is available to handle the media	startMedia	400 (BadRequest)	Unavailable Resource	No Media Resource Available
Media cannot be stopped because the specified stream ID does not exist	stopMedia	400 (BadRequest)	InvalidResourceID	Unknown Media Stream ID: <streamID>