MiaRec

SOAP API
Programmer Guide

Revision 1.1 (2014-11-15)

Contents

1 Introduction

This guide describes MiaRec SOAP Application Programming Interface (API).

The document is written for applications developers.

2 Getting started

This section describes what you need to do and what you need to know before you begin programming to this API.

2.1 Configuration of MiaRec SOAP API

Configuration of MiaRec SOAP API is available through MiaRec.ini configuration file (located in C:\Program Files\MiaRec Business\Bin).

The configuration file is a standard text file with INI format. The format is:

```
[Section Name]
Key Name=Value String
# This line is a comment
; This is a comment too
```

Comments are marked with a hash (#) or a semicolon (;) at the beginning of a line.

MiaRec reads configuration file during service start or upon a request from Telnet admin interface.

Web Services related configuration is placed in the following sections of MiaRec.ini file:

- [API::SOAP]
- [API::User::<login>]

Example of the configuration file content:

```
[API::SOAP]
Port = 6084
LoginRequired = yes

LogEnable = no
LogFileReceived = log/soap_{datetime}_{ip}_{port}_recv.log
LogFileSent = log/soap_{datetime}_{ip}_{port}_sent.log
LogFileSuffix = yyyyMMdd_hhmmss

[API::User::user1]
Password = secret1
IpAddress = 127.0.0.1
ControlledPhones = 20%
Permissions.ModifyRecordingFilters = no
Permissions.MuteRecording = yes
Permissions.OnDemandRecording = yes

[API::User::user2]
Password = secret2
IpAddress = 10.0.0.1
ControlledPhones = 200;201;202
Permissions.ModifyRecordingFilters = yes
Permissions.MuteRecording = yes
Permissions.OnDemandRecording = yes
```

Section [API::SOAP]

- **Port = SOAP_PORT**
  Default is 6084
  MiaRec application listens on this port for incoming connections from client applications.

- **LoginRequired = [yes | no]**
  Default is yes
  If "yes" then anonymous requests are forbidden and client application should authenticate itself with login/password. The login/password and other per-user settings are defined in section [API::User::<login-name>].
  During processing of the initial request, the application will need to provide credentials in the HTTP header. Credentials are passed in the HTTP authorization header using basic authentication semantics.

- **LogEnable = [yes | no]**
  Default is no
  When enabled, MiaRec application writes log into specified files for troubleshooting purposes.

- **LogFileReceived = PATH_TO_FILE**
  Default is log/soap_{datetime}_{ip}_{port}_recv.log
  Location of log file for received SOAP messages. This parameter supports placeholders {datetime}, {ip}, {port}, which are replaced with the date/time of received message, clients ip-address and port respectively.

- **LogFileSent = PATH_TO_FILE**
  Default is log/soap_{datetime}_{ip}_{port}_sent.log
  Location of log file for sent SOAP messages. This parameter supports placeholders {datetime}, {ip}, {port}, which are replaced with the date/time of sent message, client's ip-address and port respectively.

Section [API::User::<login-name>]

This section specifies login/password, which should be used by client application during communication with MiaRec web services. Additionally it allows to restrict the operations, which can be performed by particular client application.

- **[API::User::<login-name>]**
  Each web services login account should be placed into unique section [API::User::<login-name>], where *<login-name>* is a unique name used to log in to web services API.
  For example, two accounts with logins 'app1' and 'app2' and passwords '1234' and '5678' respectively are defined as:

```
[API::User::app1]
Password = 1234

[API::User::app2]
Password = 5678
```

- **Password = PASSWORD**
  The password for this the web services account.

- **IpAddress = x.x.x.x**
  Ip-address, from which the client has permissions to request web services.
  Possible values:

Value / Format	Description
any	any ip-address
x.x.x.x	ip-address value, for example, 10.0.0.2
x.x.x.x/y	ip-address mask, for example, 192.168.0.1/24
x.x.x.x/z.z.z.z	ip-address mask, for example, 192.168.0.1/255.255.255.0
x.x.x.x;m.m.m.m	multiple ip-addresses (or ip with mask), for example, 10.0.0.2;10.0.0.3

Examples:

```
IpAddress = any
IpAddress = 127.0.0.0/24
IpAddress = 192.168.0.5
IpAddress = 192.168.0.5;192.168.0.6
IpAddress = 192.168.0.0/24
IpAddress = 192.168.0.0/255.255.255.0
IpAddress = 192.168.0.0/24;10.0.0.0/24
```

- **ControlledPhones = LIST_OF_PHONES**
  List of phones, which can be controlled by client. Semicolon (;) character is used for listing multiple phones. For example:

```
ControlledPhones=100;101;102
```

The following mask symbols are supported:

Symbol	Description
_	Underscore symbol means one character. For example, mask 2_1 will match 201, 211, 251 etc.
%	Percentage symbol means zero or more characters. For example, mask 2%1 will match 21, 201, 2001, 2000001

Examples:

```
ControlledPhones = %
ControlledPhones = 200
ControlledPhones = 200;201;202
ControlledPhones = 20_
ControlledPhones = 2%
ControlledPhones = 2%;3%
```

- **Permissions.<permission-name> = [no | yes]**
  Specifies permissions for client application.
  Supported list of permissions:

Permission name	Description
ModifyRecordingFilters	Access API methods, which modify recording filters
MuteRecording	Access API methods, which mute/unmute recording
OnDemandRecording	Access API methods for triggering on-demand recording

3 Writing a client application

3.1 Obtaining the WSDL document(s) and generating the stubs

The following URL's must be used to retrieve the WSDL from the MiaRec server:

http://ServerName:6084/?wsdl

The page retrieved should be saved as a WSDL file on the application development machine. These files can be compiled into client application stub code with a language specific SOAP toolkit's WSDL compile/code generator.

3.2 Request and response framework

Your application will need to be able to:

- make service requests
- parse service responses
- catch exceptions

This section describes what requests, responses and faults are, and how to use them.

Service requests

A client application sends service requests to the MiaRec server. Examples of a request are "add new recording filter" and "mute recording" etc.

To make a request, the application calls a method on the client side service stub that was generated by their SOAP platform. The client side SOAP framework then sends a SOAP request to the service. Each request is processed by the service. Once the request is completed, the service will either send a positive response or a SOAP fault.

Service responses

Each service request is acknowledged with a service response (positive acknowledgement). The results of the request are reported in the service response.

In some cases, response to a request may result in an error. These errors are indicated through SOAP faults.

Service responses will be returned in the form of objects by the service request method on the client side stubs generated by your SOAP framework.

SOAP faults

Each service request may generate an exception, referred to as a SOAP fault. An exception indicates that the service request has failed for some reason.

Each exception will include a description of what happened and the appropriate response to a received exception will vary according to what the fault was. It will be up to the application as to how it chooses to handle such messages.

4 SOAP API

4.1 Methods

Method appGetVersion

The **appGetVersion** method returns version of MiaRec application.

Syntax (in C#):

```
string appGetVersion();
```

Return Value:

If the function succeds, it returns boolean TRUE value.

If the function fails, SOAP exception is thrown.

Example (in C#):

```
try
{
    string version = service.appGetVersion();
    Console.WriteLine(version)
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Example output:

```
MiaRec(Duxoft) Version(3.0.557) Build(Jul 14 2010, 16:03:09)
```

Method configReload

The **configReload** method forces MiaRec application to re-read configuration from MiaRec.ini configuration file.

Syntax (in C#):

```
void configReload();
```

Return Value:

The method doesn't return any value. But may throw SOAP Exception, if execution fails.

Example (in C#):

```
try
{
    service.configReload();
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Method recordingMuteByPhone

The **recordingMuteByPhone** method mutes/unmutes recording for specified phone number. This operation has effect only on the currently recorded call. It has no effect on consecutive calls.

Syntax (in C#):

```
bool recordingMuteByPhone(
    string phoneNumber,
    bool mute
);
```

Parameters:

Name of parameter	Type	Description
phoneNumber	string	Extension of the phone, which active call will be muted/unmuted
mute	bool	Mute recording when TRUE. Otherwise, unmute recording.

Return Value:

If the function succeeds, it returns boolean TRUE value.

If the function fails, then SOAP Exception is thrown.

Exceptions:

Exception name	Description
NotFoundException	There are no active calls for specified phone number
InvalidParameterException	Phone number is empty

Example (in C#):

```
try
{
    service.recordingMuteByPhone("12456", true);
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
```

```
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Method recordingMuteByCall

The recordingMuteByCall method mutes/unmutes recording of particular call, identified by ID.

Syntax (in C#):

```
bool recordingMuteByCall(
    string callId,
    bool mute
);
```

Parameters:

Name of parameter	Type	Description
callId	string	Unique ID of call to mute/unmute
mute	bool	Mute recording when TRUE. Otherwise, unmute recording.

Return Value:

If the function succeeds, it returns boolean TRUE value.

If the function fails, then SOAP Exception is thrown.

Exceptions:

Exception name	Description
NotFoundException	Call with such ID is not found or it is not active anymore
InvalidParameterException	Call ID is empty

Example (in C#):

```
try
{
    service.recordingMuteByCall("0015172d2a7c10190dee980e1eaa129d", true);
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (Exception ex)
{
```

```
    Console.WriteLine(ex.Message)
}
```

Method recordingGetMuteState

The recordingGetMuteState method returns mute state for particular call.

Syntax (in C#):

```
bool recordingGetMuteState(
    string callId
);
```

Parameters:

Name of parameter	Type	Description
callId	string	Unique ID of call to mute/unmute

Return Value:

If the function succeeds, it returns boolean TRUE value.

If the function fails, then SOAP Exception is thrown.

Exceptions:

Exception name	Description
NotFoundException	Call with such ID is not found or it is not active anymore
InvalidParameterException	Call ID is empty

Example (in C#):

```
try
{
    bool is_muted = service.recordingGetMuteState("0015172d2a7c10190dee980e1eaa129d");
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Method recordingOnDemandTriggerByCall

The **recordingOnDemandTriggerByCall** method triggers on-demand recording. It allows to mark the call as "keep" or "discard". If marked "discard", then the call will be automatically deleted upon completion. This method should be called before call completes. If it is called multiple times, then only the last method call has action.

Syntax (in C#):

```
bool recordingOnDemandTriggerByCall(
    string callId,
    bool keepRecording
);
```

Parameters:

Name of parameter	Type	Description
callId	string	Unique ID of call
keepRecording	bool	Keep recording or discard. When call is marked "discard" then it will be automatically deleted upon call completion. The on-demand recording state may be changed anytime during a call.

Return Value:

If the function succeeds, it returns boolean TRUE value.

If the function fails, then SOAP Exception is thrown.

Exceptions:

Exception name	Description
NotFoundException	Call with such ID is not found or it is not active anymore
InvalidParameterException	Call ID is empty
InvalidCallStateException	This call does not support on-demand triggering. The call is configured as "100%" recording rather than "on-demand" recording.

Example (in C#):

```
try
{
    service.recordingOnDemandTriggerByCall("0015172d2a7c10190dee980e1eaa129d", true);
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Method callGetState

The **callGetState** method returns state of particular call.

Syntax (in C#):

```
CallStateInfo recordingFilterLoad(
    string callId
);
```

Parameters:

Name of parameter	Type	Description
callId	string	Unique ID of call

Return Value:

If the function succeeds, the return value is the call state object of type **CallStateInfo**.

If the function fails, then SOAP Exception is thrown.

Exceptions:

Exception name	Description
NotFoundException	Call with such ID is not found or it is not active anymore
InvalidParameterException	Call ID is empty

Example (in C#):

```
try
{
    MiaRec::CallStateInfo state = service.recordingFilterLoad("0015172d2a7c10190dee980e1eaa129d");

    Console.WriteLine("Call State:       " + state.callstate);
    Console.WriteLine("Recording State:  " + state.recordingstate);
    Console.WriteLine("On-Demand State:  " + state.ondemandstate);
    Console.WriteLine("Duration:         " + state.duration);
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (Exception ex)
```

```
{
    Console.WriteLine(ex.Message)
}
```

Example output:

```
Call State:       4
Recording State:  10
On-Demand State:  0
Duration:         PT0H1M45
```

Method licenseGetInfo

The **licenseGetInfo** method returns license information.

Syntax (in C#):

```
LicenseInfo licenseGetInfo();
```

Return Value:

The return value is the license information object of type **LicenseInfo**.

Example (in C#):

```
try
{
    MiaRec::LicenseInfo licenseInfo = service.licenseGetInfo();

    Console.WriteLine("Trial expiration Date:        " + licenseInfo.expire);
    Console.WriteLine("Maintenance Expiration Date:  " + licenseInfo.maintenance);
    Console.WriteLine("Computer ID:                  " + licenseInfo.computerid);
    Console.WriteLine("Dongle ID:                    " + licenseInfo.dongleid);

     MiaRec::License[] licenses = licenseInfo.licenses;
     if (licenses != null)
     {
         for (int j = 0; j < licenses.Length; j++)
         {
             Console.WriteLine(licenses[j].name + ": " + licenses[j].total);
         }
     }
}
catch (FaultException<MiaRec::InvalidParameterException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (FaultException<MiaRec::NotFoundException> ex)
{
    Console.WriteLine("Failed: " + ex.Detail.Description);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message)
}
```

Example output:

```
Trial expiration Date:        2015-02-01
Maintenance Expiration Date:  2016-01-01
Computer ID:                  6BB4C8B045F2861778D2EF3868ACD54A130DF5D16DFA7D2
Dongle ID:                    1234567:1
Recording Seats: 100
Monitoring Seats: 5
```

4.2 Objects

Object CallStateInfo

The **CallStateInfo** object represents a state of call.

Syntax (in C#):

```
class CallStateInfo
{
    public int callstate;
    public int recordingstate;
    public int ondemandstate;
    public string duration;
};
```

Fields:

Name of field	Type	Description
callstate	int	1 – INITIATED (First phase of a call. The caller sent invitation to callee) 2 – ACCEPTED (The caller received invitation and confirmed it) 3 – ALERTING (The callee started ringin) 4 – CONNECTED (The call was answered. Media is started in both directions) 5 – DISCONNECTING (The call was initiated for disconnecting by one of parties) 6 – DISCONNECTED (The call was completed) 7 – HOLD (The call was put on hold) 8 – TRANSFERRED (The call was transferred to third party) 9 – DELETED (The call was deleted)
recordingstate	int	10 – ACTIVE (Call is being recorded now) 20 – LICENSE_OVERUSE (This call cannot be played back due to license overuse issue). 30 – STOPPED (Call recording is completed) 40 – IGNORED (This call is not recorded because of recording filters) 50 – FILE_ERROR (This call is not recorded due to file storage issues)

ondemandstate	int	0 – DISABLED (This call doesn't support on-demand triggering) 1 – KEEP_RECORDING (On-demand trigger is received. The call will not be deleted). 2 – WAITING_TRIGGER (Waiting for on-demand trigger. If not triggered, then the call will be deleted automatically upon completion).
duration	string	Call duration in ISO8601 format.

Example (in C#)

See example of method **callGetState()**.

Object LicenseInfo

The **LicenseInfo** object represents an information of installed licenses.

Syntax (in C#):

```
class LicenseInfo
{
    public DateTime expire;
    public DateTime maintenance;
    public string computerid;
    public string dongleid;
    public License licenses[];
};
```

Fields:

Name of field	Type	Description
expire	DateTime	Date of trial license expiration if available.
maintenance	DateTime	Date of maintenance expiration.
computerid	string	Unique Computer Id used for licensing
dongleid	string	ID of MiaRec USB dongle used for licensing
licenses[]	array of License	Installed licenses

Example (in C#)

See example of method **licenseGetInfo()**.

Object License

The **License** object represents an individual license with name and total counter.

Syntax (in C#):

```
class RecordingFilterCustomParameter
{
    public string name;
    public int total;
};
```

Fields:

Name of field	Type	Description
name	string	License name
total	int	Total number of licenses.

Example (in C#)

See example of method **licenseGetInfo()**.

4.3 Exceptions

Exception InvalidParameterException

The **InvalidParameterException** exception means that one of method parameters is invalid.

See more detailed description inside related method definition.

Fields:

Name of field	Type	Description
Description	string	Human-redable description of the error.

Exception InvalidSyntaxException

The **InvalidSyntaxException** exception means that syntax of one of method's parameters is invalid.

See more detailed description inside related method definition.

Fields:

Name of field	Type	Description
Description	string	Human-redable description of the error.

Exception NotFoundException

The **NotFoundException** exception means that the requested object cannot be found.

See more detailed description inside related method definition.

Fields:

Name of field	Type	Description
Description	string	Human-redable description of the error.

Exception InvalidCallStateException

The **InvalidCallStateException** exception means that the requested operation cannot be executed for the current calls state, for example, on-demand trigger is discarded because call is configured for 100% recording rather than on-demand recording..

See more detailed description inside related method definition.

Fields:

Name of field	Type	Description
Description	string	Human-redable description of the error.