CentovaCast
XML API Documentation

Table of Contents

1. Introduction

To facilitate interaction between CentovaCast and third-party software applications, CentovaCast provides a simple XML automation API. This manual documents the calling conventions and functionality provided the API.

This document is provided as a courtesy to CentovaCast clients who wish to integrate CentovaCast with their existing business operations. It is intended for qualified software developers and assumes that the reader has some level of experience with HTTP, XML, and software application development in general. We regret that we cannot provide support or assistance with the CentovaCast API or CentovaCast integration as part of our CentovaCast support services.

2. API Protocol

The CentovaCast automation API makes use of a very simple XML-over-HTTP protocol to accept and respond to requests. API methods are called via a standard HTTP POST request to the `api.php` script in the Centova Cast web root, in which an XML request packet is provided as POST data. CentovaCast responds to API requests by returning an XML response packet as the payload of the HTTP response.

2.1. XML Request Packet Structure

An XML request packet always begins with a standard XML header, followed by a single `<centovacast>` element. Inside the `<centovacast>` element, a single `<request>` element identifies the packet as a request packet. The `<request>` element must include both a `class` attribute indicating the API method class (described under API Method Classes below), as well as a `method` attribute indicating the API method to be called. All API methods available are described in detail under their respective headings below.

Within the `<request>` element, one or more additional elements may be provided as arguments to the API method. The name of each argument element must correspond to the name of an argument accepted by the requested API method, and the contents of each argument element must specify the value of each argument.

A typical XML request packet might look something like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<centovacast>
     <request class="system" method="info">
          <password>secret</password>
          <username>jdoe</username>
     </request>
</centovacast>
```

The packet above would indicate a request to the *info* method of the *system* class. Two arguments, *username* and *password*, are provided with values *jdoe* and *secret* respectively.

Note that while the character encoding is specified by convention in the XML preamble, Centova Cast *always* expects UTF-8 character encoding in request packets regardless of the encoding specified in the XML preamble.

2.2. XML Response Packet Structure

An XML response packet always begins with a standard XML header, followed by a single `<centovacast>` element with a `version` attribute indicating the CentovaCast version.

Inside the `<centovacast>` element, a single `<response>` element identifies the packet as a response packet. The `<response>` element always includes a `type` attribute indicating the type of result: `success` if the request was successful, or `error` if an error occurred.

Within the `<response>` element, a single `<message>` attribute always contains a textual description of the result of the request.

The `<response>` element may also include a `<data>` element containing result data generated by the request. The `<data>` element will contain zero or more `<row>` elements, each containing zero or more `<field>` elements. Each `<field>` element will include a `name` attribute specifying the name of the field, and will encapsulate the value for the field.

A typical XML response packet might look something like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<centovacast version="1.0.0">
     <response type="success">
          <message>Check complete</message>
          <data>
               <row>
                    <field name="username">jdoe</field>
                    <field name="state">up</field>
                    <field name="expected">up</field>
               </row>
          </data>
     </response>
</centovacast>
```

The response packet above would indicate that a request was successfully processed by CentovaCast version *1.0.0*. One result row was generated by the request, which contained fields entitled *username*, *state*, and *expected*, whose values were *jdoe*, *up*, and *up*, respectively.

3. API Method Classes

API methods are broken down into two classes: *system* and *server*. System methods generally correspond to tasks that would be performed by the administrator, and which would pertain to the overall management and administration of CentovaCast. Server methods correspond to tasks involving a single streaming server account.

The methods provided by each class are described under their respective headings below.

4. The System Class

Because they operate on a server-wide basis, all methods of the system class (except where otherwise noted) require a `password` argument containing the CentovaCast administrator password in addition to the arguments noted for each method below. Without the administrator password, all methods of the system class will return an authentication failure error.

The sections below describe the methods available under the system class.

4.1. Provision Account

Provisions a new client streaming server account in CentovaCast. Note that as of Centova Cast v2.2, account templates can and should be used for detailed configuration of new accounts. While the use of account templates is technically optional for user accounts, account templates *must* be used if a reseller account is being created as no other mechanism is provided to distinguish the account type.

4.1.1. Method

`provision`

4.1.2. Arguments

`hostname`	Specifies the hostname for the stream. This hostname should resolve to the IP address specified by the *ipaddress* argument. Example: `radio.example.com`
`ipaddress`	Specifies the IP address on which the streaming server should listen. This IP address must of course be configured on the server on which CentovaCast will be running. Example: `10.42.128.3`

`port`	Specifies the port number on which the streaming server should listen. This port must not already be in use by other CentovaCast streaming servers or other applications running on the server. Use `auto` to have Centova Cast select the next available port automatically. Example: `8000`
`maxclients`	Specifies the maximum number of listeners that may simultaneously tune in to this stream at any given time. Example: `10`
`username`	Specifies the username for this stream. This will be used to allow the client to login to CentovaCast. Example: `jdoe`
`adminpassword`	Specifies the password for this stream. This will be used both to administer the streaming server itself, and to allow the client to login to CentovaCast. Example: `secret`
`sourcepassword`	Specifies the source password for this stream. This will be used to allow streaming sources to connect to the streaming server and begin broadcasting. Example: `secret`
`maxbitrate`	Specifies the maximum bit rate for this stream, in kilobits per second (kbps). Note that some streaming servers (notably *IceCast*) do not enforce this setting, but it must still be specified. Example: `128`
`transferlimit`	Specifies the maximum monthly data transfer for this stream, in megabytes (MB). If you do not wish to impose a limit, specify `unlimited`. Example: `1000`
`diskquota`	Specifies the maximum disk space for this stream, in megabytes (MB). If you do not wish to impose a limit, specify `unlimited`. Example: `100`
`title`	Specifies the title for the stream. This will be displayed by listeners' media players when they tune into the stream. Example: `XYZ Widgets Streaming Radio`

`organization` Specifies the company/organization or client to whom this stream belongs.

Example: `XYZ Widgets Inc.`

`genre` Specifies the genre of the stream.

Example: `Rock`

This feature requires Centova Cast v2.0.1 or better.

`email` Specifies the client's E-mail address. CentovaCast will automatically send notifications to this address when necessary.

Example: `xyzwidgets@example.com`

`url` Specifies the URL to the web site associated with this stream (if any).

Example: `http://xyzwidgets.example.com`

`usesource` Specifies whether or not the stream uses a server-side streaming source.

Possible values include:
`0` – Use of server-side streaming source is prohibited (off-site source *must* be provided)
`1` – Use of server-side streaming source is permitted, and enabled by default
`2` – Use of server-side streaming source is permitted, but disabled by default

`introfile` Specifies the path and filename to the introduction audio file for this stream, relative to the streaming host's base directory. This may be left blank to specify no introduction audio file.

Example: `var/spool/sounds/introduction.mp3`

`fallbackfile` Specifies the path and filename to the fallback audio file for this stream, relative to the streaming host's base directory. This may be left blank to specify no introduction audio file.

Example: `var/spool/sounds/fallback.mp3`

`autorebuildlist` Specifies whether or not the playlist should be rebuilt from the stream's server-side media library every time the stream is started or restarted. This has no effect if the `usesource` argument is set to `0`.

Possible values include:
`0` – Do not automatically rebuild the playlist (unless no playlist exists)
`1` – Automatically rebuild the playlist

`autostart`

Specifies whether or not the stream should automatically be started after provisioning. Note that this option will *only* be used if the `usesource` option is set to `0` or `2`. (If server-side source, or "autodj" support is enabled, the stream cannot be started until the client has uploaded some media.)

Possible values include:
`0` – Do not automatically start the stream after provisioning. This is the default.
`1` – Automatically start the stream after provisioning if `usesource` is set to `0` or `2`.

This feature requires Centova Cast v2.0.1 or better.

`timezone`

Specifies the local time zone for the stream, in hours relative to UTC (GMT), used to ensure that the playlist scheduler uses times that make sense to the client. Leave blank or specify `auto` to use the server's time zone.

Examples:
`-8` – GMT -08:00 (Vancouver)
`0` – GMT (London)
`3` – GMT +03:00 (Moscow)

This feature requires Centova Cast v2.0.1 or better.

`allowproxy`

Specifies whether or not the stream should be permitted to use the port-80 web proxy.

Possible values include:
`0` – Disallow access to the port-80 proxy.
`1` – Allow access to the port-80 proxy.

This feature requires Centova Cast v2.1.4 or better.

`charset`

Specifies the character set for the stream.

Examples:
`ISO-8859-1` – Use the Latin-1 character set.

This feature requires Centova Cast v2.1.4 or better.

`servertype`

Specifies the streaming server type for the stream. The selected server type must be installed on the server and enabled in Centova Cast.

Examples:

	`ShoutCast` – Use ShoutCast DNAS. `IceCast` – Use IceCast. This feature requires Centova Cast v2.2.0 or better.
`servertype`	Specifies the streaming source type for the stream. The selected source type must be installed on the server and enabled in Centova Cast. Examples: `icescc` – Use ices-cc. `sctrans` – Use sc_trans This feature requires Centova Cast v2.2.0 or better.
`template`	Specifies the name of the account template to use for this account. The account template must exist in Centova Cast. Examples: `mytemplate` – Use the template named "mytemplate". This feature requires Centova Cast v2.2.0 or better.

4.1.3. Return Value

A result type of `success` is returned if the streaming server account was created successfully, otherwise `failure` is returned. No data is returned by this method.

4.2. Remove Account

Removes an existing client streaming server account from CentovaCast. The complete account (including all settings, logs, and any source media) will be permanently deleted.

4.2.1. Method

`terminate`

4.2.2. Arguments

`username`	Specifies the username of the stream to remove. Example: `jdoe`

4.2.3. Return Value

A result type of `success` is returned if the streaming server account was removed successfully, otherwise `failure` is returned. No data is returned by this method.

4.3. Set Account Status

Changes the status of an existing client streaming server account in CentovaCast.

4.3.1. Method

`setstatus`

4.3.2. Arguments

`username`	Specifies the username of the stream to update. Example: `jdoe`
`status`	Specifies the new status for the stream. Possible values include: `disabled` – Disables the stream. While disabled, the administrator will not be permitted to log in to CentovaCast, and the streaming server will remain offline. The stream will be shut down if it is online. `enabled` – Enables the stream. The stream will not be automatically be started even if it was up prior to being disabled.

4.3.3. Return Value

A result type of `success` is returned if the streaming server account was updated successfully, otherwise `failure` is returned. No data is returned by this method.

4.4. Get Account State

Returns the state (up or down) of one or more CentovaCast streaming server accounts. This can be used to monitor streams to see if any have crashed. (Note that CentovaCast's cron job automatically monitors and restarts crashed streaming servers as well.)

4.4.1. Method

`info`

4.4.2. Arguments

`username`	Specifies the username of the stream to check. To return information about *all* streaming server accounts, use the special value `all`. Example: `jdoe`

4.4.3. Return Value

A result type of `success` is returned if the list of accounts was retrieved successfully, otherwise `failure` is returned.

Zero or more result rows will be returned by this method, containing the following fields:

`username`	Indicates the username of the streaming server account that was tested.
`state`	Indicates the actual state of the streaming server for the account. Possible values include: `up` – The streaming server is online. `down` – The streaming server is offline.
`expected`	Indicates the expected state of the streaming server for the account. Possible values include: `up` – The streaming server is supposed to be online. `down` – The streaming server is supposed to be offline.
`sourcestate`	Indicates the actual state of the server-side streaming source for the account (if enabled). Possible values include: `up` – The streaming source is online. `down` – The streaming source is offline.
`sourceexpected`	Indicates the expected state of the server-side streaming source for the account (if enabled). Possible values include: `up` – The streaming source is supposed to be online. `down` – The streaming source is supposed to be offline.

5. The Server Class

Because they operate on a per-account basis, all methods of the server class (except where otherwise noted) require a `username` argument specifying the username of the streaming server account to be manipulated, and a `password` argument containing the password for the account, in addition to the arguments noted for each method below.

If desired, the password argument may be replaced with the word `admin`, followed by a pipe character (|) followed by the CentovaCast administrator password. For example, if the CentovaCast administrator password is `secret`, it is acceptable to specify `admin|secret` as the password for any streaming server account.

The sections below describe the methods available under the server class.

5.1. Reconfigure Account

Updates the settings for an existing client streaming server account in CentovaCast.

5.1.1. Method

`reconfigure`

5.1.2. Arguments

`hostname`	Specifies the hostname for the stream. This hostname should resolve to the IP address specified by the *ipaddress* argument. Example: `radio.example.com`
`ipaddress`	Specifies the IP address on which the streaming server should listen. This IP address must of course be configured on the server on which CentovaCast will be running. Example: `10.42.128.3`
`port`	Specifies the port number on which the streaming server should listen. This port must not already be in use by other CentovaCast streaming servers or other applications running on the server. Example: `8000`

`maxclients`	Specifies the maximum number of listeners that may simultaneously tune in to this stream at any given time. Example: `10`
`sourcepassword`	Specifies the source password for this stream. This will be used to allow streaming sources to connect to the streaming server and begin broadcasting. Example: `secret`
`maxbitrate`	Specifies the maximum bit rate for this stream, in kilobits per second (kbps). Note that some streaming servers (notably *IceCast*) do not enforce this setting, but it must still be specified. Example: `128`
`transferlimit`	Specifies the maximum monthly data transfer for this stream, in megabytes (MB). Example: `1000`
`diskquota`	Specifies the maximum disk space for this stream, in megabytes (MB). Example: `100`
`title`	Specifies the title for the stream. This will be displayed by listeners' media players when they tune into the stream. Example: `XYZ Widgets Streaming Radio`
`organization`	Specifies the company/organization or client to whom this stream belongs. Example: `XYZ Widgets Inc.`
`email`	Specifies the client's E-mail address. CentovaCast will automatically send notifications to this address when necessary. Example: `xyzwidgets@example.com`
`url`	Specifies the URL to the web site associated with this stream (if any). Example: `http://xyzwidgets.example.com`
`usesource`	Specifies whether or not the stream uses a server-side streaming source. Possible values include: `0` – Use of server-side streaming source is prohibited (off-site source *must* be

	provided) 1 – Use of server-side streaming source is permitted, and enabled by default 2 – Use of server-side streaming source is permitted, but disabled by default
introfile	Specifies the path and filename to the introduction audio file for this stream, relative to the streaming host's base directory. This may be left blank to specify no introduction audio file. Example: var/spool/sounds/introduction.mp3
fallbackfile	Specifies the path and filename to the fallback audio file for this stream, relative to the streaming host's base directory. This may be left blank to specify no introduction audio file. Example: var/spool/sounds/fallback.mp3
autorebuildlist	Specifies whether or not the playlist should be rebuilt from the stream's server-side media library every time the stream is started or restarted. This has no effect if the usesource argument is set to 0. Possible values include: 0 – Do not automatically rebuild the playlist (unless no playlist exists) 1 – Automatically rebuild the playlist

5.1.3. Return Value

A result type of success is returned if the streaming server account was updated successfully, otherwise failure is returned. No data is returned by this method.

5.2. Start Server

Starts a streaming server for a CentovaCast client account. If server-side streaming source support is enabled, the streaming source is started as well.

5.2.1. Method

start

5.2.2. Arguments

None (aside from the username and password arguments required by all server class methods).

5.2.3. Return Value

A result type of success is returned if the streaming server account was started successfully,

otherwise `failure` is returned. No data is returned by this method.

5.3. Stop Server

Stops a streaming server for a CentovaCast client account. If server-side streaming source support is enabled, the streaming source is stopped as well.

5.3.1. Method

`stop`

5.3.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.3.3. Return Value

A result type of `success` is returned if the streaming server account was stopped successfully, otherwise `failure` is returned. No data is returned by this method.

5.4. Restart Server

Stops, then re-starts a streaming server for a CentovaCast client account. If server-side streaming source support is enabled, the streaming source is restarted as well.

5.4.1. Method

`restart`

5.4.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.4.3. Return Value

A result type of `success` is returned if the streaming server account was restarted successfully, otherwise `failure` is returned. No data is returned by this method.

5.5. Reload Server

Reloads the streaming server configuration for a CentovaCast client account. If server-side streaming source support is enabled, the configuration and playlist for the streaming source is reloaded as well.

5.5.1. Method

`reload`

5.5.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.5.3. Return Value

A result type of `success` is returned if the streaming server configuration was reloaded successfully, otherwise `failure` is returned. No data is returned by this method.

5.6. Get Account Settings

Retrieves the configuration for a CentovaCast client account. If server-side streaming source support is enabled, the configuration for the streaming source is returned as well.

5.6.1. Method

`getaccount`

5.6.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.6.3. Return Value

A result type of `success` is returned if the streaming server configuration was reloaded successfully, otherwise `failure` is returned.

Zero or more result rows will be returned by this method, containing fields corresponding to the same arguments passed to the `reconfigure` method described above. Additional fields may be returned depending on the server and streaming source application being used on the server. These fields will specify additional configuration information for the corresponding server or source application.

5.7. Get Stream Status

Retrieves status information from the streaming server for a CentovaCast client account.

5.7.1. Method

`getstatus`

5.7.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.7.3. Return Value

A result type of `success` is returned if the streaming server status information was retrieved successfully, otherwise `failure` is returned.

Zero or more result rows will be returned by this method, containing the following fields:

`mount`	Indicates the mount point on which the stream is broadcasting.
`listenercount`	Indicates the number of listeners currently tuned in to the stream.
`genre`	Indicates the genre of the stream, as provided by the streaming source.
`url`	Indicates the URL for the stream, as provided by the streaming source.
`title`	Indicates the title for the stream, as provided by the streaming source.
`currentsong`	Indicates the title (and possibly artist) of the current track being played by the streaming server.
`bitrate`	Indicates the bit rate at which the current stream is being broadcasted, as provided by the streaming source.

5.8. Get Song History

Retrieves a list of tracks that were recently broadcasted on a given CentovaCast client's streaming server.

5.8.1. Method

`getsongs`

5.8.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.8.3. Return Value

A result type of `success` is returned if the recent song information was retrieved successfully, otherwise `failure` is returned.

Zero or more result rows will be returned by this method, containing the following fields:

`title`	Indicates the title (and possibly artist) of the track.

`time`	Indicates the time at which the track was played.

5.9. Manage Playlists

Provides access to the playlist operations for a CentovaCast client account's server-side streaming source (also sometimes referred to as "auto-DJ"). This method will only work with accounts for which server-side streaming source support is enabled.

5.9.1. Method

`playlist`

5.9.2. Arguments

`action`	Specifies the playlist action to perform. Possible values include: `activate` — Activates the selected playlist. Either `playlist` or `playlistname` must also be specified as arguments indicating the playlist to activate. `deactivate` — Deactivates the selected playlist. Either `playlist` or `playlistname` must also be specified as arguments indicating the playlist to activate. `add` — Adds tracks to the selected playlist. Either `playlist` or `playlistname` must be specified as arguments indicating the playlist to receive the tracks, and one of `trackname`, `trackpath`, `albumname`, or `artistname` must be specified indicating the tracks to add. `remove` — Removes tracks from the selected playlist. Either `playlist` or `playlistname` must be specified as arguments indicating the playlist from which the tracks should be removed, and one of `trackname`, `trackpath`, `albumname`, or `artistname` must be specified indicating the tracks to remove.
`playlist`	Specifies the ID number of the playlist on which `action` should be performed.
`playlistname`	Specifies a keyword matching the name of the playlist on which `action` should be performed. This is a more convenient alternative to specifying the playlist ID number in `playlist`.

`trackname`	Specifies a keyword matching the name of the track(s) which should be added or removed.
`trackpath`	Specifies a keyword matching the filesystem path of the track(s) which should be added or removed. Note that these tracks must already exist in the media library.
`albumname`	Specifies a keyword matching the name of the album(s) whose tracks should be added or removed.
`artistname`	Specifies a keyword matching the name of the artists(s) whose tracks should be added or removed.

5.9.3. Return Value

A result type of `success` is returned if the playlist was rebuilt successfully, otherwise `failure` is returned.

5.10. Update Media Library

Updates the media library for a CentovaCast client account's server-side streaming source (also sometimes referred to as “auto-DJ”). The media library must be re-indexed every time a client uploads new media before the new media will appear in the media library.

5.10.1. Method

`reindex`

5.10.2. Arguments

`intoplaylist`	Specifies the ID number of a playlist to which any newly-imported tracks should be added.
`intoplaylistname`	Specifies a keyword matching the name of a playlist to which any newly-imported tracks should be added. This is a more convenient alternative to the `intoplaylist` argument above.

Note that the above two arguments are optional, and are intended to allow new tracks to be easily identified by placing them automatically into a known playlist. After re-indexing, all new tracks will always appear in the media library regardless of whether either of the above two arguments are specified.

5.10.3. Return Value

A result type of `success` is returned if the media was imported successfully, otherwise `failure` is returned.

5.11. Advance to Next Song

Skips to the next song in the playlist for a CentovaCast client account's server-side streaming source. This method will only work with accounts for which server-side streaming source support is enabled.

5.11.1. Method

`nextsong`

5.11.2. Arguments

None (aside from the `username` and `password` arguments required by all server class methods).

5.11.3. Return Value

A result type of `success` is returned if the song was skipped successfully, otherwise `failure` is returned. No data is returned by this method.

5.12. Activate/Deactivate Streaming Source

Starts or stops the server-side streaming source (or "auto-DJ") for a Centova Cast client account. This can be used to stop the auto-DJ before a live stream commences, and start it afterward.

5.12.1. Method

`switchsource`

5.12.2. Arguments

`state`	Specifies the new state for the streaming source. Possible values include: `up` – Activates the streaming source. `down` – Deactivates the streaming source.

5.12.3. Return Value

A result type of `success` is returned if the source state was changed successfully, otherwise `failure` is returned.

5.13. Copy File

Copies a file into a subdirectory of the `var/spool/` directory for a CentovaCast client account. This is typically used to install media files (such as the stream introduction or fallback files, or media for use by the server-side streaming source) to a given account with the appropriate ownership and privileges.

5.13.1. Method

`copyfile`

5.13.2. Arguments

`sourcefile`	Specifies the *absolute* path and filename to a file on the server to be copied. This file *must* reside on the same server on which CentovaCast is running. Example: `/tmp/myfile.mp3`
`destfile`	Specifies the *relative* path and filename (relative to the client account's `var/spool/` directory) to which the source file should be copied. Example: `media/myfile.mp3` This example would copy the file to `var/spool/media/myfile.mp3` under the client account's root directory.

5.13.3. Return Value

A result type of `success` is returned if the file was copied successfully, otherwise `failure` is returned. No data is returned by this method.

6. Credits

CentovaCast

http://www.centova.com