1. Ask your account manager for your personal publisher ID.
2. Integrate the code on every game page.
3. Connect your game id's to the walkthrough code.

Getting started

Copy the following snippet within your web document at the location where you want the video to appear. Make sure you set values of the following mandatory properties gameId, title and publisherId.
All three properties should be string values (plain text), without the < and > characters. Make sure you put the values between double quotes ("). You can find definitions of all these properties at the bottom of this document.

The video will only appear when these three values are valid.

Take note that it can take a few days until Tubia knows/processed which video should be displayed for each game. For testing purposes you could also force a video by temporary setting the title value to *Run 3* , which is a game already known by Tubia.

```
<div id="videoContainer"></div>
   <script>
        var playerInstance;

        var settings = {
            container: "videoContainer",
            gameId: "[your game id]", //it is an unique value
            publisherId: "[your publisher id]",
            width: "100%",
            height: "480px",
            title: "[Game Title]", //you have to type your game title here we need for
matching game video
            category: "[Game Category]", // game category
```

Advanced example

Use the following example snippet if you want more control in what the video player does, or want to hook onto an event.

```
<div id="videoContainer"></div>
   <script>
        var playerInstance;

        var settings = {
            container: "videoContainer",
            gameId: "[your game id]", //it is an unique value
            publisherId: "379ab3d446b047a9afce7b9628d69c7b",
            width: "100%",
            height: "480px",
            title: "[Game Title]", //you have to type your game title here we need for
matching game video
            category: "[Game Category]", // game category
            langCode: "en-us",
            autoplay: "no",
            onFound: function (data) {
                console.log(data);
            },
            onError: function (data) {
                console.log(data);
            },
            onReady:function(player){
                // Keep player instance to control it later like play or pause player
                // available controls:
                //
                // player.play() or player.play([seek time]); // You can jump a time
(second *optional) of video.
                // player.pause();
                // player.setVolume(0.5); // Value should be between 0.0 and 1.0

                console.log(player);

                playerInstance = player; // Set player instance
             }
        };
        (function(i,s,o,g,r,a,m){

i['VooxeVideo']=r;i[r]=i[r]||function(){(i[r].q=i[r].q||[]).push(arguments)};i[r].l=1*new
Date();a=s.createElement(o);m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode
.insertBefore(a, m);
        })(window, document, 'script', '//video-static.vooxe.com/libs/gd/gd.js',
'gdPlayer');
        gdPlayer(settings);

        //
```

```
        // Player Controls
        //
        function play() {
            if (playerInstance!=null)
            {
                playerInstance.play(0);
            }
        }
        function pause() {
            if (playerInstance!=null)
            {
                playerInstance.pause();
            }
        }
        function setvolume() {
            if (playerInstance!=null)
            {
                playerInstance.setVolume(0.5);
            }
        }

    </script>
```

Documentation

All the Tubia configuration settings are represented as properties of the **settings** object. The following properties are supported:

container

Type: `[String]`

Default: `empty string`

Mandatory: `yes`

The id of the html element where the Tubia video will be embedded into.

publisherId

Type: `String`

Default: `empty string`

Mandatory: `yes`

The publisher id will be given to you by your Tubia account manager. The player won't retrieve video's, without a proper id.

gameId

Type: `[String]`
Default: `empty string`
Mandatory: `yes`

The game identification code coming from your games database. An example of how we use this id is to cross reference between any of your other sites so we can figure out the proper video for your game, even when the game title differs. Think of the case where multi language websites have game titles in different languages. In those cases we can use the id, instead of looking for a video using the game title.

title

Type: `[String]`
Default: `empty string`
Mandatory: `yes`

The title will be used to find a walkthrough for your game in the Tubia database.

category

Type: `[String]`
Default: `empty string`
Mandatory: `no`

You can supply the name of the main category the game belongs to on your website. Only one category name should be supplied.

width

Type: `[String]`
Default: `100%`
Mandatory: `no`

This property defines the width of the player. It requires a CSS value. Meaning you can use pixels, view width, percentages, em's, etc...

height

Type: `[String]`

Default: `480px`

Mandatory: `no`

This property defines the height of the player. It requires a CSS value. Meaning you can use pixels, view height, percentages, em's, etc...

langCode

Type: `[String]`

Default: `en-us`

Mandatory: `no`

This property can be used to set the correct language and country of all the strings within the Tubia player. Make sure it is a lowercased valid ISO value for language (https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes) and country (https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2).

autoplay

Type: `[String]`

Default: `no`

Mandatory: `no`

Setting this property value to yes will make the video start playing right away. Take note that this will likely not work on mobile devices, simply because mobile browsers require a user interaction before allowing the video to start playing.

onFound

Type: `[Function (data)]`

Mandatory: `no`

An optional callback function, which will be called after a video has been found by Tubia. The data argument is provided directly from Tubia. You could use this callback to display/enable a "see walkthrough" button for example.

onError

Type: `[Function (error)]`
Mandatory: `no`

An optional callback function, which will be called if Tubia returns an error. The error argument is provided directly from Tubia and contains additional error information.

onReady

Type: `[Function (player)]`
Mandatory: `no`

An optional callback function, which will be called after Tubia is fully ready to play a video for your user. The player argument is the HTML5 video object, which can be used to pause, resume etc. as shown in the advanced example.

Release History

See the CHANGELOG.

License