Vanguard Datastream – User Guide

info@uniblox.com
www.uniblox.com

Table of contents

About

Vanguard Datastream is a data management tool for Unity. It has the following features:

- Save and load data in a custom Editor Window or Inspector.
- Save and load data to and from disk at runtime.
- Upload and download data from and to an FTP server.
- Serialize a Dictionary<> object, which isn't supported in Unity out of the box.
- Full source code included; fully documented.

Setup

Introduction

This section will detail all of the necessary steps to get Vanguard Datastream up and running inside your Unity project.

Adding Vanguard Datastream to your project

After purchasing Vanguard Datastream on the Unity Asset Store, press the "Download" button. Vanguard Datastream will be downloaded and the "Import Unity Package" window will open. Make sure that everything is selected, and press "Import".

Managers

Vanguard Datastream utilizes a "manager" script, in the form of a singleton; FTPManager. This manager requires you to have a scene in your project which needs to remain loaded at all times and which contains this manager. To speed things up, we've included a "Management" scene with the plugin, located under *Vanguard > Datastream > Scenes*. Simply load this scene (and keep it loaded) to complete this part of the setup. If your project already includes such a scene, or if you want to create the scene yourself, you can also

Note: To make scene management easier, check out our other plugin, Theatre, in the Unity Asset Store.

FTPManager

This manager makes all of this plugin's FTP related features possible. To set it up, simply open up the inspector of the FTPManager in your "Management" scene which was set up in the previous section, and fill it in. Hover over an element in the inspector to see that element's tooltip.

Demos

Introduction

Vanguard Datastream includes demos with profusely documented code which show off Vanguard Datastream's features and can help use these features yourself. You can find them under *Vanguard > Datastream > Demo*.

EditorStorage example

This demo showcases the ability to save and load data inside a custom Editor Window by using EditorStorage. The demo includes a basic Editor Window which displays both an integer value and a string value. These values get loaded from, and get saved to, an EditorStorage. This means that any change you make to either of these values will persist when the Editor Window is closed or even if Unity itself is closed.

Runtime Example

This demo showcases both the ability to save and load data from and to disk by using RuntimeStorage and the ability to upload and download files to and fro an FTP server. The demo includes an InputField and buttons which allow you to test these features. The recommended use of this demo is to first set up FTPManager (see the previous page for details on how to do this) and to build and run this demo outside of Unity. This way you will get to test each of the features showcased in this demo to their fullest extent.

EditorStorage

Introduction

With EditorStorage, you are able to save and load data from custom Editor Windows and Inspectors by storing the data externally. This data will be kept even if Unity is closed.

Usage

To use the EditorStorage, follow these steps:

1. Create a new instance of EditorStorage. In the Project window, navigate to *Create > Vanguard > Datastream* and click *EditorStorage*. Both the name and place of the instance are irrelevant to usability.
2. In the inspector of the created instance, set the reference key. This key will be used to access the instance from a custom Editor Window or Inspector.
3. In your custom Editor Window or Inspector, get a reference to your EditorStorage instance by calling *EditorStorage.GetInstance()*.
4. For each value that you want to store in the EditorStorage, create a StoredItem<> field and initialize it.
5. To load a value from the EditorStorage, call *Load()* on the reference to the EditorStorage and supply a reference to the StoredItem<>.
6. For the values to be saved, call *SetObjects()* on the reference to the EditorStorage and supply all the StoredItem<> instances that should be saved. **Important: Any StoredItem<> instance that isn't supplied in this call will instantly be removed from the EditorStorage.**

RuntimeStorage

Introduction

With RuntimeStorage, you are able to keep runtime data in between sessions by saving the data to, and loading the data from the platform's persistent data path.

Usage

To use the RuntimeStorage, follow these steps:

1. Create a new instance of RuntimeStorage. In the Project window, navigate to *Create > Vanguard > Datastream* and click *RuntimeStorage*. Both the name and place of the instance are irrelevant to usability.
2. In your script, add a RuntimeStorage field make sure that this field gets serialized by making it public or by using the SerializeField attribute.
3. In the inspector of your C# script, reference the RuntimeStorage instance which you created earlier.
4. For each value that you want to store in the RuntimeStorage, create a StoredItem<> field and initialize it.
5. To load a value from the RuntimeStorage, call *Load()* on the reference to the RuntimeStorage and supply a reference to the StoredItem<>.
6. For the values to be saved, call *SetObjects()* on the reference to the RuntimeStorage and supply all the StoredItem<> instances that should be saved. **Important: Any StoredItem<> instance that isn't supplied in this call will instantly be removed from the RuntimeStorage.**

FTP

Introduction

This section is meant to explain how to save and load both files and objects to and from an FTP server using Vanguard Datastream, and assumes that you have both an FTP server ready for use and have set up the FTPManager as explained earlier in the documentation.

Upload an object to the FTP server

To upload an object to the FTP server, call *MemoryUtility.FTPUpload()*. This method requires you to pass the object which should be uploaded, the destination folder on the server and the name with which the file will be uploaded to the server. The method returns a bool indicating whether or not the operation was successful.

Download an object from the FTP server

To download an object from the FTP server, call *MemoryUtility.FTPDownload()*. This method requires you to pass the folder at which the file containing the object is located, the name of the file and an out parameter with the type bool which will be set to true or false depending on whether or not the operation was successful.

Uploading a file from disk to the FTP server

To upload a file to the FTP server, call *FileUtility.FTPUpload()*. This method requires you to pass the file info, the destination folder on the server and a bool to indicate whether or not the directory should be made. The method returns a bool indicating whether or not the operation was successful.

Downloading a file to disk from the FTP server

To download a file from the FTP server, call *FileUtility.FTPUpload()*. This method requires you to pass the folder on the server where the file is located, the name of the file, the path on disk where the file will be saved to and an out parameter with the type bool which will be set to true or false depending on whether or not the operation was successful.

Outro

Thank you for choosing Vanguard Datastream! We hope that everything is clear to you, and that you'll have a great time using this product. If it has left a positive impression, we'd like to ask you to rate this product on the Unity Asset Store. This way, more people will be able to find it!

Any questions, feature requests, complaints or compliments? You can reach us at info@uniblox.com.