DataSaving System

Thread-safe Unity save system with encryption, compression, and multi-slot support.

Setup

1. Attach `SetupWizard` component to any GameObject
2. Configure compression/encryption settings
3. Enable auto-slot generation if needed

```
using SaveLoadSystem; // Basic usage DataManager.Save("playerName", "John");
DataManager.Save("level", 42); string name = DataManager.Load<string>("playerName"); int level =
DataManager.Load<int>("level");
```

Core API

`DataManager.Save(string key, object value)`

```
void Save(string key, object value)
```

Saves value synchronously. Thread-safe.

```
DataManager.Save("score", 1500); DataManager.Save("position", transform.position);
```

`DataManager.SaveAsync(string key, object value)`

```
async Task SaveAsync(string key, object value)
```

Saves value asynchronously.

```
await DataManager.SaveAsync("largeData", bigObject);
```

`DataManager.Load<T>(string key)`

```
T Load<T>(string key)
```

Loads value synchronously. Returns `default(T)` if not found.

```
int score = DataManager.Load<int>("score"); Vector3 pos = DataManager.Load<Vector3>("position");
```

`DataManager.DeleteValue(string key)`

```
void DeleteValue(string key)
```

Deletes saved value immediately.

```
DataManager.DeleteValue("tempData");
```

DataManager.DeleteValueAsync(string key)

```
async Task DeleteValueAsync(string key)
```

Deletes saved value asynchronously.

```
await DataManager.DeleteValueAsync("tempData");
```

Save Slot Management

DataManager.CreateSaveSlot(string name)

```
void CreateSaveSlot(string name)
```

```
DataManager.CreateSaveSlot("Player1");
```

DataManager.SetActiveSlot(string name)

```
void SetActiveSlot(string name)
```

```
DataManager.SetActiveSlot("Player1");
```

DataManager.DeleteSaveSlot(string name)

```
void DeleteSaveSlot(string name)
```

```
DataManager.DeleteSaveSlot("OldSave");
```

Permanently deletes all data in the slot.

DataManager.RenameSaveSlot(string oldName, string newName)

```
void RenameSaveSlot(string oldName, string newName)
```

```
DataManager.RenameSaveSlot("TempSave", "MainSave");
```

DataManager.GetSaveSlots()

```
List<SaveSlot> GetSaveSlots()
```

```
var slots = DataManager.GetSaveSlots(); foreach(var slot in slots) { Debug.Log(slot.Name); }
```

DataManager.DoesSlotExist(string name)

```
bool DoesSlotExist(string name)
```

```
if(DataManager.DoesSlotExist("Player1")) { DataManager.SetActiveSlot("Player1"); }
```

DataManager.DeleteAllSaveSlots()

```
void DeleteAllSaveSlots()
```

```
DataManager.DeleteAllSaveSlots();
```

Deletes ALL save data permanently.

Supported Unity Types

Type	Example
Vector2, Vector3, Vector4	`Save("pos", transform.position)`
Vector2Int, Vector3Int	`Save("gridPos", gridPosition)`
Quaternion	`Save("rotation", transform.rotation)`
Color, Color32	`Save("color", Color.red)`
Rect, RectInt	`Save("bounds", rect)`
Bounds, BoundsInt	`Save("area", bounds)`
Matrix4x4	`Save("matrix", matrix)`
Ray, Ray2D	`Save("ray", ray)`
Plane	`Save("plane", plane)`
AnimationCurve	`Save("curve", curve)`
Gradient	`Save("gradient", gradient)`

Configuration

Compression

```
DataManager.CompressionType = Compression.None; // No compression DataManager.CompressionType = Compression.GZIP; // GZIP compression
```

Encryption

```
DataManager.EncryptionType = Encryption.None; // No encryption DataManager.EncryptionType = Encryption.AES; // AES encryption
```

Logging

```
CustomLogger.CurrentLogLevel = LogLevel.None; // Errors only CustomLogger.CurrentLogLevel = LogLevel.Minimal; // Warnings + errors CustomLogger.CurrentLogLevel = LogLevel.Detailed; // All messages
```

SaveSlot Methods

slot.GetLastModified()

```
DateTime GetLastModified()
```

```
DateTime lastModified = slot.GetLastModified();
```

slot.GetFileSize()

```
long GetFileSize()
```

```
long size = slot.GetFileSize();
```

slot.LoadMetadata()

```
SaveMetadata LoadMetadata()
```

```
SaveMetadata metadata = slot.LoadMetadata(); DateTime saveTime = metadata.saveTime; Compression compression = metadata.compression; Encryption encryption = metadata.encryption;
```

Thread Safety

All operations are thread-safe using:

- SemaphoreSlim for file access coordination
- Lock objects for cache management
- Thread-safe collections

File Structure

Save files are stored in `Application.persistentDataPath/Saveslots/`:

- `SlotName.saveslot` - Main save data
- `SlotName_KEY.key` - Encryption key (if encryption enabled)
- `SlotName_METADATA.metadata` - Save metadata