ICloud Keychain for iOS

Keychain provides functionality for securely storing, retrieving, and deleting sensitive data in the iOS Keychain with optional iCloud synchronization. This allows apps to persist information such as user credentials, authentication tokens, and other sensitive data in a secure encrypted container that remains available even after app uninstallation.

Key Features

- **Secure Storage**: Leverages iOS Keychain Services for encryption and security
- **Persistence**: Data persists even when the app is uninstalled
- **iCloud Synchronization**: Optional syncing of Keychain items across user's devices
- **Thread Safety**: All operations are thread-safe
- **Comprehensive Error Handling**: Detailed error messages and status codes

API Reference

Keychain Class

Properties

Property	Type	Description
`IsPlatformSupported`	`bool`	Indicates whether Keychain operations are available on the current platform (iOS only).
`IsICloudAvailable`	`bool`	Indicates whether iCloud Keychain is available and properly configured on the current device. The result is cached for performance (60 seconds by default).

Methods

Method	Parameters	Return Type	Description
`Set`	`string key, string value, bool iCloud`	`Result`	Always saves data in the local device Keychain first. If `iCloud` is set to `true` and iCloud Keychain is available, also saves a second copy to iCloud for synchronization across devices. This dual-storage approach ensures data availability even when iCloud is temporarily unavailable or network connectivity is limited.
`Get`	`string key, bool iCloud`	`Result<string>`	When `iCloud` is `true` and iCloud Keychain is available, first attempts to retrieve the value from iCloud Keychain, which may have the most recent data synchronized from other devices. If not found in iCloud or if iCloud is unavailable, falls back to searching in the local Keychain. When `iCloud` is `false`, only searches in the local Keychain.
`Delete`	`string key, bool iCloud`	`Result`	Always attempts to delete the value from the local Keychain first. When `iCloud` is `true` and iCloud Keychain is available, also deletes the value from iCloud, ensuring removal across all synchronized devices. When `iCloud` is `false`, only the local copy is deleted while any iCloud copy remains intact.

Result Class

Provides information about the outcome of Keychain operations.

Properties

Property	Type	Description
`Success`	`bool`	Indicates whether the operation was successful.
`ErrorCode`	`int`	The error code associated with the operation. A value of 0 indicates success, any other value indicates an error.
`Message`	`string`	A human-readable localized error message if the operation failed. May contain a success message when the operation succeeds.

Methods

Method	Parameters	Return Type	Description
`ToString`	none	`string`	Returns a string representation of the Result including Success, ErrorCode, and Message values.

Result\<T> Class

Extends `Result` to include a returned value from operations that retrieve data.

Properties

Property	Type	Description
`Value`	`T`	The value returned by the operation. Will be default(T) if the operation failed or no value was found.

Methods

Method	Parameters	Return Type	Description
`ToString`	none	`string`	Returns a string representation of the Result\<T> including Value, Success, ErrorCode, and Message.

Usage Examples

Basic Operations

```
// Save a value
var setResult = Keychain.Set("user_password", "my_secure_password", false);
if (setResult.Success)
{
    Debug.Log("Password saved successfully");
}
else
{
    Debug.LogError($"Failed to save password: {setResult.Message}");
}

// Retrieve a value
var getResult = Keychain.Get("user_password", false);
if (getResult.Success)
{
    var password = getResult.Value;
```

```
    Debug.Log($"Retrieved password: {password}");
}
else
{
    Debug.LogError($"Failed to retrieve password: {getResult.Message}");
}

// Delete a value
var deleteResult = Keychain.Delete("user_password", false);
if (deleteResult.Success)
{
    Debug.Log("Password deleted successfully");
}
else
{
    Debug.LogError($"Failed to delete password: {deleteResult.Message}");
}
```

Using iCloud Synchronization

```
// Check if iCloud Keychain is available
if (Keychain.IsICloudAvailable)
{
    Debug.Log("iCloud Keychain is available");

    // Save a value with iCloud synchronization
    var setResult = Keychain.Set("cloud_token", "my_synchronized_token", true);
    if (setResult.Success)
    {
        Debug.Log("Token saved with iCloud synchronization");
    }

    // Retrieve a value with iCloud support
    var getResult = Keychain.Get("cloud_token", true);
    if (getResult.Success)
    {
        string token = getResult.Value;
        Debug.Log($"Retrieved synchronized token: {token}");
    }
}
else
{
    Debug.Log("iCloud Keychain is not available, using local storage only");

    // Operations will still work, but only using local storage
    Keychain.Set("cloud_token", "my_local_token", true); // iCloud param is ignored
}
```

Storing Complex Objects

```
// Create a user data object
var userData = new UserData
{
    Username = "player123",
    Level = 42,
    LastLogin = DateTime.Now
};

// Serialize to JSON
var json = JsonUtility.ToJson(userData);

// Save to Keychain
var setResult = Keychain.Set("user_data", json, true);
if (setResult.Success)
{
    Debug.Log("User data saved successfully");
}

// Retrieve and deserialize
var getResult = Keychain.Get("user_data", true);
if (getResult.Success)
{
    UserData retrievedData = JsonUtility.FromJson<UserData>(getResult.Value);
    Debug.Log($"Retrieved user: {retrievedData.Username}, Level: {retrievedData.Level}");
}
```

Handling Different Storage Strategies

```
// For sensitive data you want available across devices
Keychain.Set("auth_token", token, true); // Store both locally and in iCloud

// For device-specific settings
Keychain.Set("device_id", deviceId, false); // Store locally only

// Get synchronized data (checks iCloud first, then local)
var result = Keychain.Get("auth_token", true);
if (result.Success) {
    // Use result.Value
}

// Get only from this device (checks local storage only)
var localResult = Keychain.Get("device_id", false);
if (localResult.Success) {
    // Use localResult.Value
}

// Deleting synchronized data (removes from both local and iCloud)
Keychain.Delete("auth_token", true);
```

```
// Deleting only from this device (removes only local copy)
Keychain.Delete("temp_cache", false);
```

Error Handling

For robust applications, always check the `Success` property of the `Result` returned by each operation:

```
Result<string> result = Keychain.Get("api_token", true);
if (result.Success)
{
    // Use the value
    string token = result.Value;
    // ...
}
else
{
    // Handle the error
    switch (result.ErrorCode)
    {
        case -25300: // Item not found
            Debug.Log("Token not found, requesting new one...");
            // Request new token
            break;
        case -25301: // iCloud not available
            Debug.Log("iCloud not available, using local storage");
            // Fall back to local-only mode
            break;
        default:
            Debug.LogError($"Error {result.ErrorCode}: {result.Message}");
            break;
    }
}
```

Requirements for iCloud Synchronization

For iCloud Keychain synchronization to work:

1. User must be signed into iCloud on the use device.
2. iCloud Keychain must be enabled in device settings on the user device.
3. App must have proper entitlements configured (automatically handled by the plugin).
    a. **iCloud Keychain Synchronization**
    This entitlement enables the app to use iCloud services, which is necessary for Keychain synchronization.
    `com.apple.developer.icloud-services`
    b. **Keychain Sharing**
    Allows your app to share Keychain items across multiple apps from the same development team.
    `keychain-access-groups`
4. User must have sufficient iCloud storage space.

Size Limitations

The maximum size for Keychain items is generally around 2MB, but it's recommended to keep values small for better performance.

Troubleshooting

Common Issues

Issue	Possible Solution
"Item not found" error	Verify the key name and check if the item was previously saved
iCloud synchronization not working	Verify user is logged into iCloud and has iCloud Keychain enabled
"No permission" error	Ensure app has proper entitlements. Plugin should handle this automatically
Data not syncing between devices	iCloud synchronization may take time. Ensure internet connectivity

Error Codes

Error Code	Name	Description
0	errSecSuccess	Operation completed successfully
-25300	errSecItemNotFound	The requested item could not be found in the keychain
-25301	errSecICloudNotAvailable	iCloud is not available or not properly configured
-25302	errSecDataConversionError	Failed to convert data to/from string format
-25303	errSecUnknown	An unknown error occurred
-25304	errSecDuplicateItem	An item with the same attributes already exists
-25305	errSecSaveFailed	Failed to save data to the keychain
-25306	errSecDeleteFailed	Failed to delete data from the keychain
-25307	errSecNoPermission	No permission to access the keychain
-25308	errSecUserCanceled	User canceled the operation
-25309	errSecNetworkFailure	Network failure during iCloud synchronization
-25310	errSecQuotaExceeded	iCloud storage quota exceeded
-34018	errSecMissingEntitlement	Missing required entitlement
-26275	errSecInteractionNotAllowed	User interaction is not allowed
-25294	errSecAuthFailed	Authorization/Authentication failed
-25292	errSecDecode	Unable to decode the provided data
-25293	errSecInvalidParam	One or more parameters passed to the function were not valid
-25291	errSecNotAvailable	The keychain is not available
-25317	errSecReadOnly	The keychain is read-only
-25243	errSecServiceNotAvailable	The service is not available
-25299	errSecWrongSecVersion	Wrong Security framework version