Secure Store static library

You can store all kind of sensitive data in it: user credentials, passwords, secret keys etc. Once stored in Keychain this information is only available to your app, other apps can't see it. Besides that, operating system makes sure this information is kept and processed securely. For example, text stored in Keychain cannot be extracted from iPhone backup or from its file system. Apple recommends storing only small amount of data in the Keychain.

REVISION HISTORY

DATE	VERSION	AUTHOR	UPDATE DESCRIPTION
July 26, 2020	1.0	Gautham Velappan	Initial Draft

KeyStore

```
public class KeyStore : SecureStore
```

A collection of helper functions for saving, retrieving, updating and deleting Keys(string) in the keychain.

- `init(accessGroup:)`
  Instantiate a KeyStore object

  Declaration
  SWIFT
  ```
  override public init(accessGroup: String? = nil)
  ```

  Parameters

accessGroup	Specify an access group that will be used to access keychain items. Access groups can be used to share keychain items between applications. When access group value is nil all application access groups are being accessed. Access group name is used by all functions: add, get, delete and update.

- `add(_:for:completion:)`
  Adds the Key(string) value in the keychain item under the given user.

  Declaration
  SWIFT
  ```
  public func add(_ value: String, for user: String, completion:
  ((Error?) -> Void)? = nil)
  ```

  Parameters

value	String to be written to the keychain.
user	The user for which the string value is stored in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `getValues(for:completion:)`
  Retrieves the Key(string) from the keychain that corresponds to the given user.

  Declaration
  SWIFT
  ```
  public func getValues(for user: String, completion: ((Error?) ->
  Void)? = nil) -> [String]?
  ```

  Parameters

user	The user that is used to read the keychain item.
completion	A closure of type Error to be executed once the request has finished.

  Return Value
  The Keys(string array) value from the keychain.

- `delete(value:for:completion:)`
  Deletes a single(provided key) or all key items specified by the given user.

  Declaration
  SWIFT
  ```
  public func delete(value: String? = nil, for user: String, completion:
  ((Error?) -> Void)? = nil)
  ```

  Parameters

value	String to be deleted from the keychain.
user	The user for which the keys are deleted in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `update(_:with:for:completion:)`
  Updates an existing key value in the keychain item under the given user.

Declaration

SWIFT

```
public func update(_ oldValue: String, with newValue: String, for
user: String, completion: ((Error?) -> Void)? = nil)
```

Parameters

oldValue	The existing Key(string) to be replaced to the keychain.
newValue	The new Key(string) to be updated to the keychain.
user	The user for which the string value is updated in the keychain.
completion	A closure of type Error to be executed once the request has finished.

CredentialStore

```
public class CredentialStore : SecureStore
```

A collection of helper functions for saving, retrieving, updating and deleting Credentials object in the keychain.

- `init(accessGroup:)`
  Instantiate a CredentialStore object

Declaration

SWIFT

```
override public init(accessGroup: String? = nil)
```

Parameters

accessGroup	Specify an access group that will be used to access keychain items. Access groups can be used to share keychain items between applications. When access group value is nil all application access groups are being accessed. Access group name is used by all functions: add, get, delete and update.

- `add(_:for:completion:)`
  Adds the Credential object in the keychain item under the given user.

Declaration

SWIFT

```
public func add(_ value: Credential, for user: String, completion:
((Error?) -> Void)? = nil)
```

Parameters

value	Credential object to be written to the keychain.
user	The user for which the string value is stored in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `getValues(for:completion:)`
  Retrieves the Credential objects from the keychain that corresponds to the given user.

  Declaration
  SWIFT
  ```
  public func getValues(for user: String, completion: ((Error?) ->
  Void)? = nil) -> [Credential]?
  ```

  Parameters

`user`	The user that is used to read the keychain item.
`completion`	A closure of type Error to be executed once the request has finished.

  Return Value
  The Credential objects(Credential array) value from the keychain.

- `delete(value:for:completion:)`
  Deletes a single(provided credential) or all credential items specified by the given user.

  Declaration
  SWIFT
  ```
  public func delete(value: Credential? = nil, for user: String,
  completion: ((Error?) -> Void)? = nil)
  ```

  Parameters

`value`	Credential to be deleted from the keychain.
`user`	The user for which the keys are deleted in the keychain.
`completion`	A closure of type Error to be executed once the request has finished.

- `update(_:with:for:completion:)`
  Updates an existing Credential object in the keychain item under the given user.

  Declaration
  SWIFT
  ```
  public func update(_ oldValue: Credential, with newValue: Credential,
  for user: String, completion: ((Error?) -> Void)? = nil)
  ```

  Parameters

oldValue	The existing Credential to be replaced to the keychain.
newValue	The new Credential to be updated to the keychain.
user	The user for which the Credential value is updated in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `add(_:for:completion:)`
  Adds the Credential object in the keychain item under the given user.

  Declaration
  SWIFT
  ```
  public func add(_ value: [String : Any], for user: String, completion:
  ((Error?) -> Void)? = nil)
  ```

  Parameters

value	Credential object to be written to the keychain.
user	The user for which the string value is stored in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `getValues(for:completion:)`
  Retrieves the Credential JSON from the keychain that corresponds to the given user.

  Declaration
  SWIFT
  ```
  public func getValues(for user: String, completion: ((Error?) ->
  Void)? = nil) -> [[String : Any]]?
  ```

  Parameters

user	The user that is used to read the keychain item.
completion	A closure of type Error to be executed once the request has finished.

  Return Value
  The Credentials (JSON array) value from the keychain.

- `delete(value:for:completion:)`
  Deletes a single(provided credential JSON) or all credential items specified by the given user.

  Declaration
  SWIFT
  ```
  public func delete(value: [String : Any]? = nil, for user: String,
  completion: ((Error?) -> Void)? = nil)
  ```

  Parameters

value	Credential JSON to be deleted from the keychain.
user	The user for which the keys are deleted in the keychain.
completion	A closure of type Error to be executed once the request has finished.

- `update(_:with:for:completion:)`
  Updates an existing Credential JSON in the keychain item under the given user.

Declaration

SWIFT

```
public func update(_ oldValue: [String : Any], with newValue: [String
: Any], for user: String, completion: ((Error?) -> Void)? = nil)
```

Parameters

oldValue	The existing Credential(JSON) to be replaced to the keychain.
newValue	The new Credential(JSON) to be updated to the keychain.
user	The user for which the Credential value is updated in the keychain.
completion	A closure of type Error to be executed once the request has finished.

DataStore

```
public class DataStore : SecureStore
```

A collection of helper functions for saving, retrieving, updating and deleting data in the keychain.

- `init(accessGroup:)`

  Instantiate a DataStore object

  Declaration

  SWIFT

  ```
  override public init(accessGroup: String? = nil)
  ```

  Parameters

`accessGroup`	Specify an access group that will be used to access keychain items. Access groups can be used to share keychain items between applications. When access group value is nil all application access groups are being accessed. Access group name is used by all functions: add, get, delete and update.

- `add(_:for:completion:)`

  Adds the data value in the keychain item under the given user.

  Declaration

  SWIFT

  ```
  public func add(_ value: Data, for user: String, completion: ((Error?)
  -> Void)? = nil)
  ```

  Parameters

`value`	Data to be written to the keychain.
`user`	The user for which the string value is stored in the keychain.
`completion`	A closure of type Error to be executed once the request has finished.

- `getValues(for:completion:)`
  Retrieves the data from the keychain that corresponds to the given user.

  Declaration
  SWIFT
  ```
  public func getValues(for user: String, completion: ((Error?) ->
  Void)? = nil) -> [Data]?
  ```

  Parameters

`user`	The user that is used to read the keychain item.
`completion`	A closure of type Error to be executed once the request has finished.

  Return Value
  The data array value from the keychain.

- `delete(value:for:completion:)`
  Deletes a single(provided data) or all data items specified by the given user.

  Declaration
  SWIFT
  ```
  public func delete(value: Data? = nil, for user: String, completion:
  ((Error?) -> Void)? = nil)
  ```

  Parameters

`value`	Data to be deleted to the keychain.
`user`	The user for which the keys are deleted in the keychain.
`completion`	A closure of type Error to be executed once the request has finished.

- `update(_:with:for:completion:)`
  Updates an existing key value in the keychain item under the given user.

  Declaration
  SWIFT
  ```swift
  public func update(_ oldValue: Data, with newValue: Data, for user:
  String, completion: ((Error?) -> Void)? = nil)
  ```

  Parameters

oldValue	The existing data to be replaced to the keychain.
newValue	The new data to be updated to the keychain.
user	The user for which the string value is updated in the keychain.
completion	A closure of type Error to be executed once the request has finished.

Credential

```
public class Credential
```

- context

  The value of the @context property MUST be an ordered set where the first item is a URI with the value https://www.w3.org/2018/credentials/v1. For reference, a copy of the base context is provided in Appendix § B. Base Context. Subsequent items in the array MUST express context information and be composed of any combination of URIs or objects. It is RECOMMENDED that each URI in the @context be one which, if dereferenced, results in a document containing machine-readable information about the @context.

  Declaration

  SWIFT

  ```
  public var context: [String]?
  ```

- type

  The value of the type property MUST be, or map to (through interpretation of the @context property), one or more URIs. If more than one URI is provided, the URIs MUST be interpreted as an unordered set. Syntactic conveniences SHOULD be used to ease developer usage. Such conveniences might include JSON-LD terms. It is RECOMMENDED that each URI in the type be one which, if dereferenced, results in a document containing machine-readable information about the type.

  Declaration

  SWIFT

  ```
  public var type: [String]?
  ```

- id

  The value of the id property MUST be a single URI. It is RECOMMENDED that the URI in the id be one which, if dereferenced, results in a document containing machine-readable information about the id.

  Declaration

  SWIFT

  ```
  public var id: String?
  ```

- issuer
  The value of the issuer property MUST be either a URI or an object containing an id property. It is RECOMMENDED that the URI in the issuer or its id be one which, if dereferenced, results in a document containing machine-readable information about the issuer that can be used to verify the information expressed in the credential.

  Declaration
  SWIFT
  ```
  public var issuer: String?
  ```
- issuanceDate
  A credential MUST have an issuanceDate property. The value of the issuanceDate property MUST be a string value of an [RFC3339] combined date and time string representing the date and time the credential becomes valid, which could be a date and time in the future. Note that this value represents the earliest point in time at which the information associated with the credentialSubject property becomes valid.

  Declaration
  SWIFT
  ```
  public var issuanceDate: String?
  ```
- expirationDate
  If present, the value of the expirationDate property MUST be a string value of an [RFC3339] combined date and time string representing the date and time the credential ceases to be valid.

  Declaration
  SWIFT
  ```
  public var expirationDate: String?
  ```
- credentialSchema
  The value of the credentialSchema property MUST be one or more data schemas that provide verifiers with enough information to determine if the provided data conforms to the provided schema. Each credentialSchema MUST specify its type (for example, JsonSchemaValidator2018), and an id property that MUST be a URI identifying the schema file. The precise contents of each data schema are determined by the specific type definition.

  Declaration
  SWIFT
  ```
  public var credentialSchema: CredentialSchema?
  ```

- credentialSubject
  The value of the credentialSubject property is defined as a set of objects that contain one or more properties that are each related to a subject of the verifiable credential. Each object MAY contain an id, as described in Section § 4.2 Identifiers.

  Declaration
  SWIFT
  ```swift
  public var credentialSubject: CredentialSubject?
  ```
- proof
  One or more cryptographic proofs that can be used to detect tampering and verify the authorship of a credential or presentation. The specific method used for an embedded proof MUST be included using the type property.

  Declaration
  SWIFT
  ```swift
  public var proof: Proof?
  ```
- rawDictionary
  A JSON representation of the Credential object. Key-Value pairs which contain all the high-level credential details

  Declaration
  SWIFT
  ```swift
  public var rawDictionary: [String : Any]?
  ```
- rawString
  A String representation of the Credential object.

  Declaration
  SWIFT
  ```swift
  public var rawString: String?
  ```

CredentialSubject

```
public class CredentialSubject
```

- `rawDictionary`
  A JSON representation of the CredentialSubject object.

  Declaration
  SWIFT
  ```
  public var rawDictionary: [String : Any]?
  ```

- `rawString`
  A String representation of the CredentialSubject object.

  Declaration
  SWIFT
  ```
  public var rawString: String?
  ```

Proof

```
public class Proof
```

- `rawDictionary`
  A JSON representation of the Proof object.

  Declaration
  SWIFT
  ```
  public var rawDictionary: [String : Any]?
  ```

- `rawString`
  A String representation of the Proof object.

  Declaration
  SWIFT
  ```
  public var rawString: String?
  ```

CredentialSchema

```
public class CredentialSchema
```

- `rawDictionary`
  A JSON representation of the CredentialSchema object.

  Declaration
  SWIFT
  ```
  public var rawDictionary: [String : Any]?
  ```

- `rawString`
  A String representation of the CredentialSchema object.

  Declaration
  SWIFT
  ```
  public var rawString: String?
  ```