URI Scheme: “ves:”

URI scheme “ves:” is defined in compliance with RFC 3986. The purpose of “ves:” scheme is to provide unique global identifiers for objects within VES Repository.

This document defines “ves:” scheme for two separate types of VES objects:

- Vaults
- Vault Items

URIs for Vaults

VES URI for Vault may take one of the following forms:

An App Vault, identified by an App Domain and an External ID. An optional *userRef* is needed to automatically create a temporary Vault Key for an App Vault that didn’t exist yet, or is used to assert that an existing App Vault belongs to the specified user:

```
“ves:” || [ “//” || { domain } || “/” || ] { externalID } || “/”
                        [ || { userRef } ]
```

A Primary Vault, identified by userRef:

```
“ves:” || [ “///” || ] “/” { userRef }
```

A Vault Key identified by an internal numeric ID. This syntax allows to access any types of Vault Keys, including specific temporary keys, lost keys, and recently deleted keys:

```
“ves:///” || { internalID } || “/” [ || { userRef } ]
```

In all statements above, “||” denotes concatenation, parts in “[“ brackets “]” are optional.

userRef is a user’s email address, with or without “<” angular quotes “>” and a personal name preceding the first angular quote, the personal name optionally in double quotes, formatted according to RFC 822, all characters after the first closing angular quote MUST be ignored. A proper “%” encoding SHOULD be applied according to RFC 3986.

Unescaped “?” and “#” should be treated as URI part separators, for compatibility with any possible future revisions.

Note: the *externalID* or *internalID* part in a Vault URI MUST be followed by “/”, even if *userRef* is not supplied. It’s important for distinguishing Vault URIs from Vault Item URIs.

URIs for Vault Items

A Vault Item can be referenced by an App Domain and an External ID:

```
“ves:” || [ “//” || { domain } || “/” || ] { externalID }
```


Jim Zubov <jz@vesvault.com>, rev. Dec 20, 2021

Or, by its internal numeric ID. This syntax allows to access internal and recently deleted Vault Items:

"ves:///" || { *internalID* }

Note: the *externalID* or *internalID* is always the last part and MUST NOT be followed by "/", for distinguishing from Vault URIs.

Examples of Valid VES URIs:

`ves://demo/myvault/`
(an existing App Vault)

`ves:myvault/`
(an existing App Vault within the default domain for the current context)

`ves://demo/myfriend/My%20Friend%20%3Cmyfriend@mydomain.com%3E`
`ves://demo/myfriend2/myfriend2@mydomain.com`
(these App Vaults will be automatically created if didn't exist)

`ves:/My%20Friend3%20%3Cmyfriend3@mydomain.com%3E%28Comment%29`
`ves:///myfriend4@mydomain.com`
(Primary Vaults)

`ves:///1234/`
(a Vault Key by internal ID)

`ves://demo/myitem`
(a Vault Item)

`ves:myItem`
(a Vault Item within the default domain for the current context)

`ves:///5678`
(a Vault Item by internal ID)

Security Considerations

VES URI identifies an encrypted object located in VES Repository, but provides no means to decrypt it. For some objects, even the encrypted data and associated metadata are protected by VES Repository access control.

The user needs to have an appropriate passphrase (VESkey) in their possession to decrypt the retrieved data, or to identify, retrieve and decrypt the keychain needed to decrypt the data. VESkey is not included in VES URI.

In certain cases, special access tokens can be given to the user that allow to retrieve and verify certain metadata, but not to decrypt the item. Such tokens are not included in the current specifications of VES URI either.

Versioning

Future versions of VES URI Specifications may specify additional features provided in a Query String, Fragment, and/or other parts of the URI, as long as they are not in conflict with the current Specifications.