

Share Transfer Module - tKey JS SDK

[@tkey/share-transfer](#)

â

The Share Transfer Module helps you transfer a share to another device/ storage.

Installationâ

- npm
- Yarn
- pnpm

npm install --save @tkey/share-transfer yarn add @tkey/share-transfer pnpm add @tkey/share-transfer

Initializationâ

```
import
{
  ShareTransferModule
}
from
"@tkey/share-transfer" ; const shareTransferModule =
new
ShareTransferModule ( ) ;
```

Usageâ

With theShareTransferModule , you have access to the multiple functions as mentioned in the type reference, however, the most important ones are:

Request a New Share from a Deviceâ

requestNewShare (userAgent :

string , availableShareIndexes :

Array < string

, callback ? :

(err ? :

ITkeyError , shareStore ? :

ShareStore)

=>

void) :

Promise < string

; * userAgent * : The user agent of the client that is requesting a new share. * availableShareIndexes * : An array of share indexes that are available for the client. * callback * : A callback function that is called when the request is complete.

Returnâ

- Promise

- : Share index of the new share.

Example[â](#)

```
const result_requestNewShare =
await ( tKey . modules . shareTransfer
as
ShareTransferModule ) . requestNewShare ( navigator . userAgent , tKey . getCurrentShareIndexes ( ) ) ;
```

Look for Requests from Another Device[â](#)

```
lookForRequests(): Promise>;
```

Return[â](#)

- Promise>
- : An array of indexes of pending requests

Example[â](#)

```
const requests =
await
( tKey . modules . shareTransfer
ShareTransferModule ) . lookForRequests ( ) ;
```

Approve request from Another Device[â](#)

```
approveRequest(encPubKeyX: string, shareStore?: ShareStore): Promise;
```

- encPubKeyX
- : The public key of the share that is being approved.
- shareStore
- : The share store that is being approved.

Example[â](#)

```
const requests =
await ( tKey . modules . shareTransfer
as
ShareTransferModule ) . lookForRequests ( ) ; let shareToShare ; try
{ shareToShare =
await ( tKey . modules . webStorage
as
WebStorageModule ) . getDeviceShare ( ) ; }
catch
( err )
{ console . error ( "No on device share found. Generating a new share" ) ; const newShare =
await tKey . generateNewShare ( ) ; shareToShare = newShare . newShareStores [ newShare . newShareIndex . toString (
"hex" ) ] ; } await ( tKey . modules . shareTransfer
as
ShareTransferModule ) . approveRequest ( requests [ 0 ] , shareToShare ) ;
```

Type Reference^â

ShareTransferModule

^â

declare

class

ShareTransferModule

implements

IModule

{ moduleName :

string ; tbSDK :

ITKeyApi ; currentEncKey :

BN ; requestStatusCheckId :

number ; requestStatusCheckInterval :

number ; constructor () ; static

refreshShareTransferMiddleware (generalStore :

unknown , oldShareStores :

ShareStoreMap , newShareStores :

ShareStoreMap) :

ShareTransferStorePointer ; setModuleReferences (tbSDK :

ITKeyApi) :

void ; setRequestStatusCheckInterval (interval :

number) :

void ; initialize () :

Promise < void

 ; requestNewShare (userAgent :

string , availableShareIndexes :

Array < string

 , callback ? :

(err ? :

ITkeyError , shareStore ? :

ShareStore)

=>

void) :

Promise < string

 ; addCustomInfoToShareRequest (encPubKeyX :

string , customInfo :

string) :

```

Promise < void
    ; lookForRequests ( ) :

Promise < Array < string
    ; approveRequest ( encPubKeyX :
string , shareStore ? :
ShareStore ) :

Promise < void
    ; approveRequestWithShareIndex ( encPubKeyX :
string , shareIndex :
string ) :

Promise < void
    ; getShareTransferStore ( ) :

Promise < ShareTransferStore
    ; setShareTransferStore ( shareTransferStore :
ShareTransferStore ) :

Promise < void
    ; startRequestStatusCheck ( encPubKeyX :
string , deleteRequestAfterCompletion :
boolean ) :

Promise < ShareStore
    ; cancelRequestStatusCheck ( ) :

Promise < void
    ; deleteShareTransferStore ( encPubKey :
string ) :

Promise < void
    ; resetShareTransferStore ( ) :

Promise < void
    ; private _cleanUpCurrentRequest ; }

```

ShareStore

[â](#)

class

ShareStore

implements

ISerializable

{ share :

Share ; polynomialID :

PolynomialID ; constructor (share :

Share , polynomialID :

PolynomialID) ; static

fromJSON (value :

StringifiedType) :

ShareStore ; toJSON () :

StringifiedType ; } interface

ISerializable

{ toJSON () :

StringifiedType ; } [Edit this page](#) [Previous Security Questions](#) [Next Share Serialization](#)