# Permits

Permits are the successor to viewing keys. With increases in efficiency made in the Shockwave update they are now the defacto viewing permission method in almost every situation.

What is a Permit?

Permits are simply a dummy transaction signed by a wallet. The wallet is able to use this signature as proof that the transactor is who they say they are. Any wallet capable of using Tendermint is able to carry out this process.

For an in-depth implementation of query permits, navigate to thesnip-24 section of the docs here . Generating a permit happens outside of the smart-contract, but checking them in your smart contract is extremely simple. Once again the Secret-Toolkit provides a package for us, aptly namedpermit . The function that checks the permit is calledvalidate shown below.

```

Copy pubfnvalidate( deps:&Extern, storage_prefix:&str, permit:&Permit, current_token_address:HumanAddr, hrp:Option<&str>, )->StdResult { letaccount_hrp=hrp.unwrap_or("secret");

if!permit.check_token(&current_token_address) { returnErr(StdError::generic_err(format!( "Permit doesn't apply to token {:?}, allowed tokens: {:?}", current_token_address.as_str(), permit .params .allowed_tokens .iter() .map(|a|a.as_str()) .collect:: <Vec<&str>>() ))); }

// Derive account from pubkey letpubkey=&permit.signature.pub_key.value;

letbase32_addr=pubkey_to_account(pubkey).0.as_slice().to_base32(); letaccount:String=bech32::encode(account_hrp,&base32_addr, Variant::Bech32).unwrap();

// Validate permit_name letpermit_name=&permit.params.permit_name; letis_permit_revoked=RevokedPermits::is_permit_revoked( &deps.storage, storage_prefix, &HumanAddr(account.clone()), permit_name, ); ifis_permit_revoked { returnErr(StdError::generic_err(format!( "Permit {:?} was revoked by account {:?}", permit_name, account.as_str() ))); }

// Validate signature, reference: https://github.com/enigmampc/SecretNetwork/blob/f591ed0cb3af28608df3bf19d6cfb733cca48100/cosmwasm/packages/wasmi-runtime/src/crypto/secp256k1.rs#L49-L82 letsigned_bytes=to_binary(&SignedPermit::from_params(&permit.params))?; letsigned_bytes_hash=sha_256(signed_bytes.as_slice());

letverified=deps .api .secp256k1_verify(&signed_bytes_hash,&permit.signature.signature.0,&pubkey.0) .map_err(|err|StdError::generic_err(err.to_string()))?;

if!verified { returnErr(StdError::generic_err( "Failed to verify signatures for the given permit", )); }

Ok(account) }

```

To use this, simply callvalidate inside of your query provided with the necessary permit variables.

When is it Best to Use Permits?

Permits can be used at any time, and are amazing for almost every permissioned viewing application. The only time it may be more efficient to use viewing keys instead is during inter-contract communication.

Last updated7 months ago On this page *What is a Permit? * When is it Best to Use Permits?

Was this helpful? Edit on GitHub Export as PDF