SOLANA

CASE STUDY

# Token Extensions on Solana

## Abstract

**Token extensions are the next generation of the Solana Program Library Token standard. Over a dozen extensions provide advanced configurable functionality, specifically designed to meet the needs of businesses with compliance obligations.**

Token extensions (TE) are about optionality and choice—and represent the next major step forward for what's possible on the Solana Network.

In this paper, we'll look in detail at token extensions. We'll see how it works and its benefits. We'll look at its wide variety of extensions and novel use cases. And finally, we'll see how token extensions can meet your project's unique needs.

## Introduction

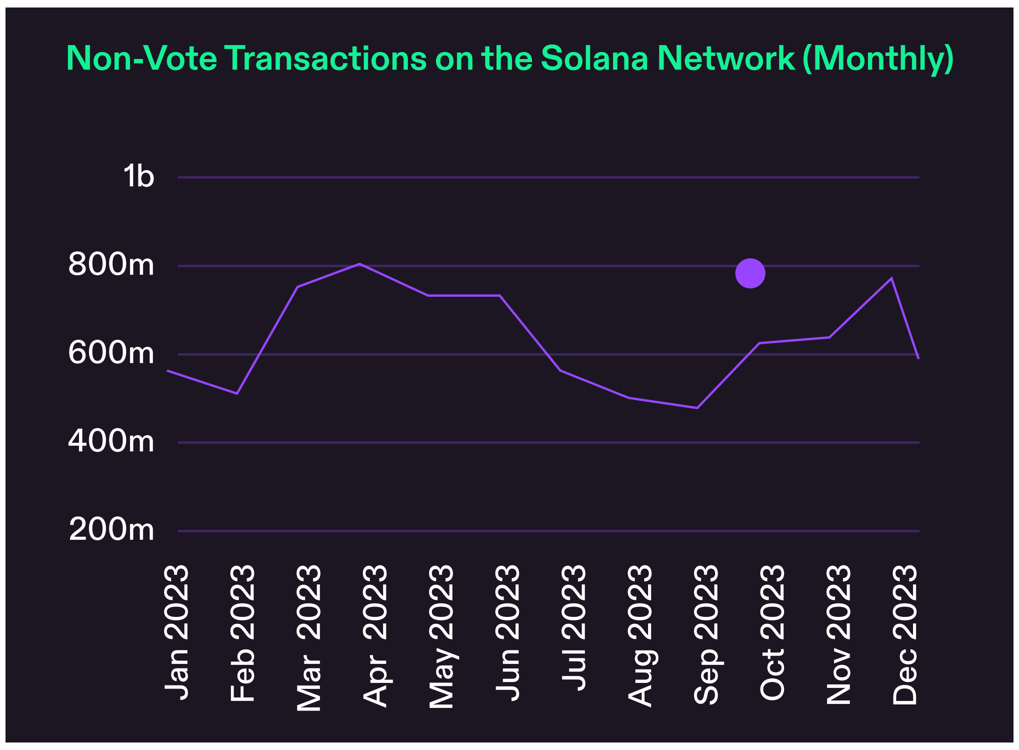
If you're part of an enterprise team that is building on blockchain—or thinking about building— then you probably already know just how difficult this task can be. You're trying to meet the unique business needs of your project, achieve the reliability and performance requirements of your enterprise, and solve the technical problems and restrictions of blockchain—all while staying on budget, on time, and compliant.

If this all sounds familiar, token extensions could be your solution. With token extensions, you can now **create advanced, regulatory-friendly, versatile blockchain tokens that are fast to build, flexible, and meet the criteria you need for your enterprise applications**—all with the scalability, security, and reliability built into the core of the Solana blockchain.

## The Original Solana Token Program

Over the past several years, Solana has become one of the most widely adopted blockchains. In Q4 2023, there was an average of 40.7 million daily user transactions processed on Solana, with more than 750 million daily transactions in December 2023 alone. This was many times more than the 35 million transactions processed on Ethereum, or the 17 million transactions on Bitcoin.

Many of these Solana transactions involved tokens created using the Token Program—a standard Solana Program Library (SPL) framework for creating and managing fungible and non-fungible tokens on Solana. Its simple interfaces have made it a powerful and straightforward way for teams to create and work with tokens.



Source: *The Block*

## What are Token Extensions?

> *Token extensions enable permissioned tokens on a permissionless environment*

Token extensions is a new token program on the Solana blockchain that enables a set of modular extensions for token issuers. These extensions are ***built into the core protocol level*** of Solana and apply to both fungible and non-fungible tokens.

With token extensions, developers can now use a set more than a dozen proven, audited extensions, that quickly add the needed advanced functionality to their tokens, such as the privacy-preserving technology of confidential transfers, new compliance frameworks such as transfer hooks, and the ability to charge fees on transfers.

You can think of the extensions as a series of options, features, and capabilities built into the newest iteration of the Solana token program. Token issuers – from game developers to stablecoin issuers – can choose to enable any combination of token extensions, gaining advanced capabilities that in some cases were previously not possible on public blockchains.

Token extensions give you native support for the enterprise features you need—without any additional tooling, vendor lock-in, or requirement to convince other teams to support your token. Nor do you need to face the complexity of developing, auditing, and deploying your own custom token contracts. With token extensions, you reduce the work needed from your engineering teams, allowing you to use your time and resources on the business problems your team is uniquely built to solve.

> ***Solana token extensions*** *provide enterprises with advanced, regulatory-friendly, versatile token functionality combined with the scalability, security, and reliability required for business-critical blockchain applications.*

## The Benefits of Token Extensions

As a major upgrade to the original SPL Token Program, token extensions offer many benefits, which include:

- **Flexibility:** By using extensions, you can customize the behavior of your tokens for your team's specific blockchain use case—and all at the base layer of your stack.
- **No additional programs needed:** Since the extensions are built into the core protocol layer, you can easily create tokens that need advanced functionality without the need for additional libraries. And you don't need to convince wallets and other protocols to support your token.
- **Reduced risk:** Using audited and well-tested extensions reduces attack vectors and helps to protect protocols and funds. With token extensions on Solana, you get enterprise-grade security and reliability.
- **Reduced testing costs:** Because the extensions are added by simply specifying the extensions in your code, the chances of defects and human error are greatly reduced, saving on testing time and costs. See how easy this is in the "Deploying a Token with Token Extensions" section below.
- **Reduced development time:** Because the extensions are uniform and reusable, the time required to develop applications using the extensions is significantly reduced. The resources saved can instead be spent on your team's expertise—implementing new blockchain use cases.
- **Faster pilots:** Faster testing, faster development, and reduced security risks drastically reduce the time for your team to create blockchain-powered projects. And since token extensions are a standard, you don't need to spend time lobbying wallets and other projects to support your custom tokens. All this means faster pilots, faster user feedback, and a faster path to product/market fit.
- **Reduced need for audits:** By using proven and battle-tested extensions, you reduce or even remove the need for expensive audits.
- **Enterprise ready:** For enterprises exploring blockchain, token extensions offer the fastest and most reliable path to harnessing the advances of blockchain. Token extensions give you the ability to customize your projects with advanced token behavior and economics. Solana gives you the scalability, low costs, enterprise-grade security, and speed you need from blockchain.

## Available Token Extensions

Now that we understand why token extensions are a major step forward for blockchain, let's take a look through the currently available extensions. All of these extensions can be used out-of-the-box by your team.

Extensions can be of two types: **mint** and **account** extensions.

Mint extensions are added on top of the original Solana Token Program and extend the abilities of tokens. Account extensions are added on top of Solana accounts and add account-related features.

Current mint extensions include:

**Confidential transfers:** Allow confidential transfers between participating users without revealing the amount of the transfer.

**Transfer fees:** Allow transfer fees to be charged on each transfer and sent to a defined account.

**Mint close authority:** Allows owners to close mint accounts and reclaim the lamports on the mint account.

**Interest-bearing tokens:** Allow setting an interest rate on a token and retrieving a record of accumulated interest for display purposes.

**Non-transferable tokens:** Allow limiting token transfers between users.

**Permanent delegate:** Allows specifying a permanent account delegate for a mint. This delegate has unlimited delegate privileges over any account for that mint, which means that it can burn or transfer any amount of tokens. Examples include the ability to seize assets from a sanctioned entity or repossess a token with an unpaid Harberger Tax.

**Transfer hook:** Allows calling specific programs with each token transfer.

**Metadata pointer:** Allows token creators to designate an address that describes the canonical (official) metadata for the token. This can even be the mint itself (see the Metadata extension below).

**Metadata:** Allows metadata to be incorporated natively into tokens through custom fields.

**Group pointer:** Allows a token creator to designate a group account (see above) that describes the mint.

**Group pointer:** Allows a token creator to designate a group account (see above) that describes the mint.

**Group:** With TE, tokens can now belong to a group. Groups can be configured with maximum size, initial size, update authority, etc.

**Member:** Describes configurations for a group member, such as group address and a member's number.

**Member pointer:** Allows the token creator to designate a member account (see above) that describes the mint.

Current account extensions include:
**Memo required on transfer:** Requires an attached memo as a message during each token transfer. This could be used for regulatory compliance, reporting, and enhanced audit trails.

**Immutable owner:** Makes it impossible to reassign ownership of an account.

**Default account state:** Freezes all new token accounts so that users must interact with the project in some way to unfreeze the accounts/tokens.

**CPI guard:** Restricts how other programs can interact with your token by prohibiting certain actions inside cross-program invocations.

**Reallocate:** Some extensions can be enabled after account creation. Reallocate allows owners in this situation to reallocate their token account to create room for more extensions.

Extensions can be mixed and matched. You can create a mint with only transfer fees, only interest-bearing tokens, or both! You can find a complete and updated list of TEs in the extension guide. To understand the power of these extensions, let's look at several in detail.

## Confidential Transfers

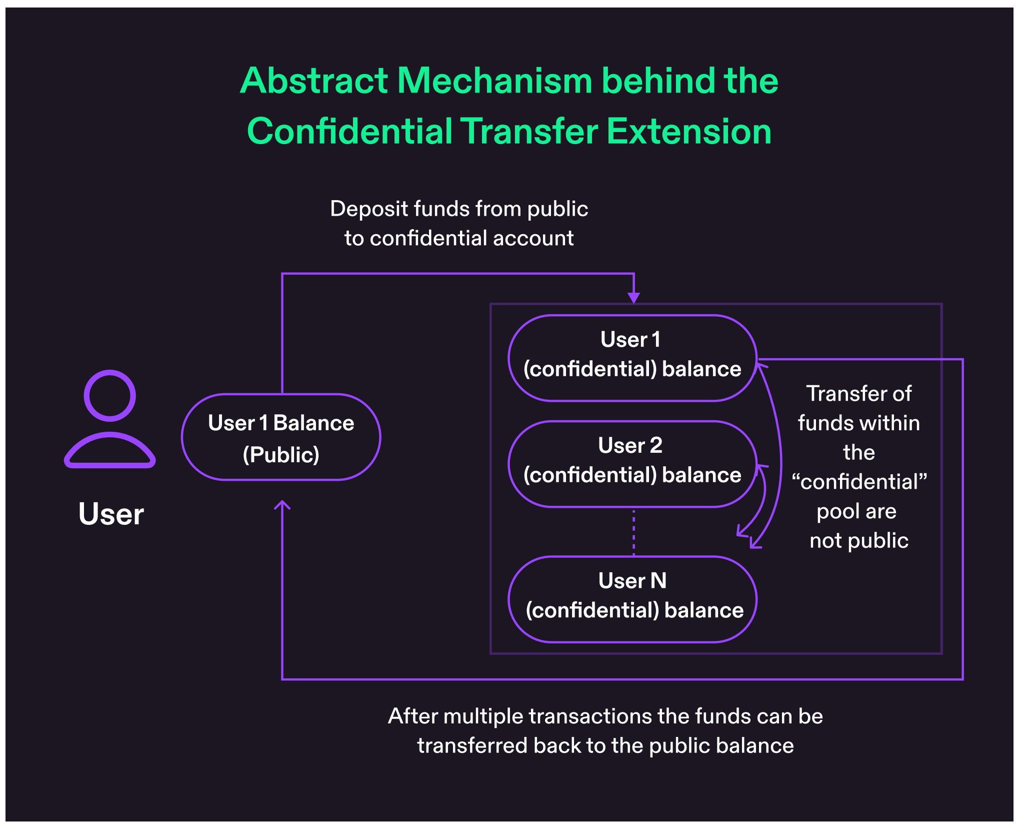
A transparent, public ledger of transactions is a foundational feature of blockchain. It increases trust in the chain, enhances security, and creates a traceable, accountable store of data.

However, this transparency can be a problem when there is a strong need for privacy. For example, a company that pays employees in USDC may want to keep their employee's salaries confidential. A trading firm may want to keep details of their OTC trades private to maintain a competitive edge.
A non-profit may want to keep its grants confidential.
And so on.

The **confidential transfer extension** allows users to securely transact without revealing the transfer amounts. With the extension, an account owner may configure their account for confidential transfers, then deposit tokens from their non-confidential holdings to the new confidential balance. Once completed, all transfers out of the confidential balance to another account that has been configured for confidential transfers are private. Transfer balances can only be seen by the source, the destination, and an optional third-party auditor.

Additionally, by using a manual approval policy, you can gate the confidential transfer functionality to only certain users, respecting the needs of different projects.

Note that the transfers themselves are not anonymous—the source, destination, and token are public, as is the history of transactions between interacting parties. However, the amount transferred between parties remains private.


Abstract Mechanism behind the Confidential Transfer Extension
Deposit funds from public to confidential account
User
User 1 Balance (Public)
User 1 (confidential) balance
User 2 (confidential) balance
User N (confidential) balance
Transfer of funds within the "confidential" pool are not public
After multiple transactions the funds can be transferred back to the public balance

Confidential transfers are one of the most impressive token extensions (powered by an underlying cryptographic protocol). They enable configurable confidentiality, hide transfer amounts from the public ledger, and support third-party accounting, all while respecting the integrity of the chain with full transparency into the source, destination, and token type.

## Transfer Hooks

The **transfer hooks extension** calls a piece of custom logic when a token is transferred.

A common use case for this extension is implementing programmatically-enforced NFT royalties—a hot topic in digital art and collectibles. Instead of current royalty solutions involving complicated proxy layers that must be known and called by wallets and marketplaces, transfer hooks allow developers to call program-defined logic before the transfer settlement.

Implementation is straightforward, as developers simply develop and deploy a program that implements the required interface and executes their business logic. Then, developers configure their token mint to use the new program.

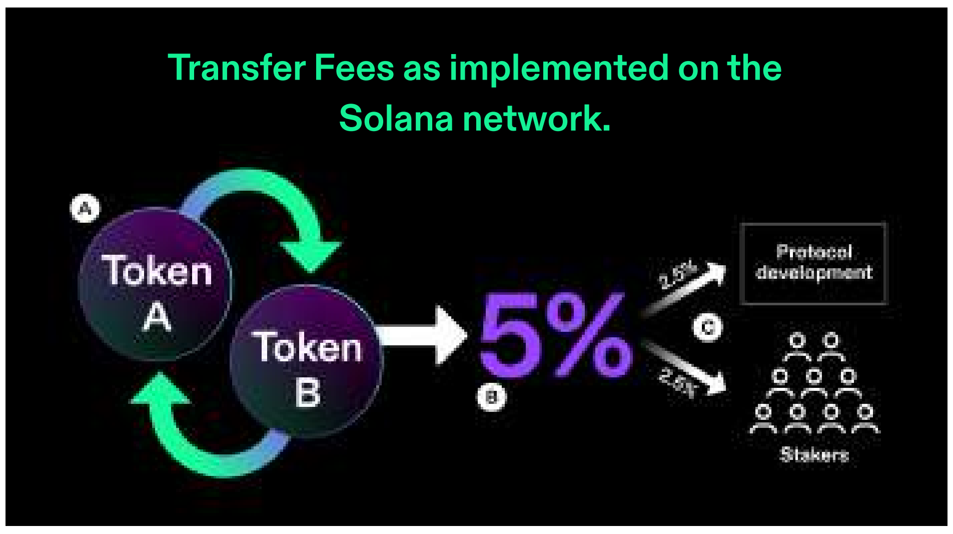
More complex control-based logic can be implemented, such as an exchange taking a specific action if a large amount of tokens is transferred between parties or if the account is on a compliance-related list. A game creator may want to take a cut of each transaction, control recipients of a transfer to guard against scammers or bad actors, or freeze game currencies if a subscription lapses.

And because transfer hooks are part of TEs, they work across all marketplaces, wallets, and apps. No manual intervention or knowledge by projects is necessary.

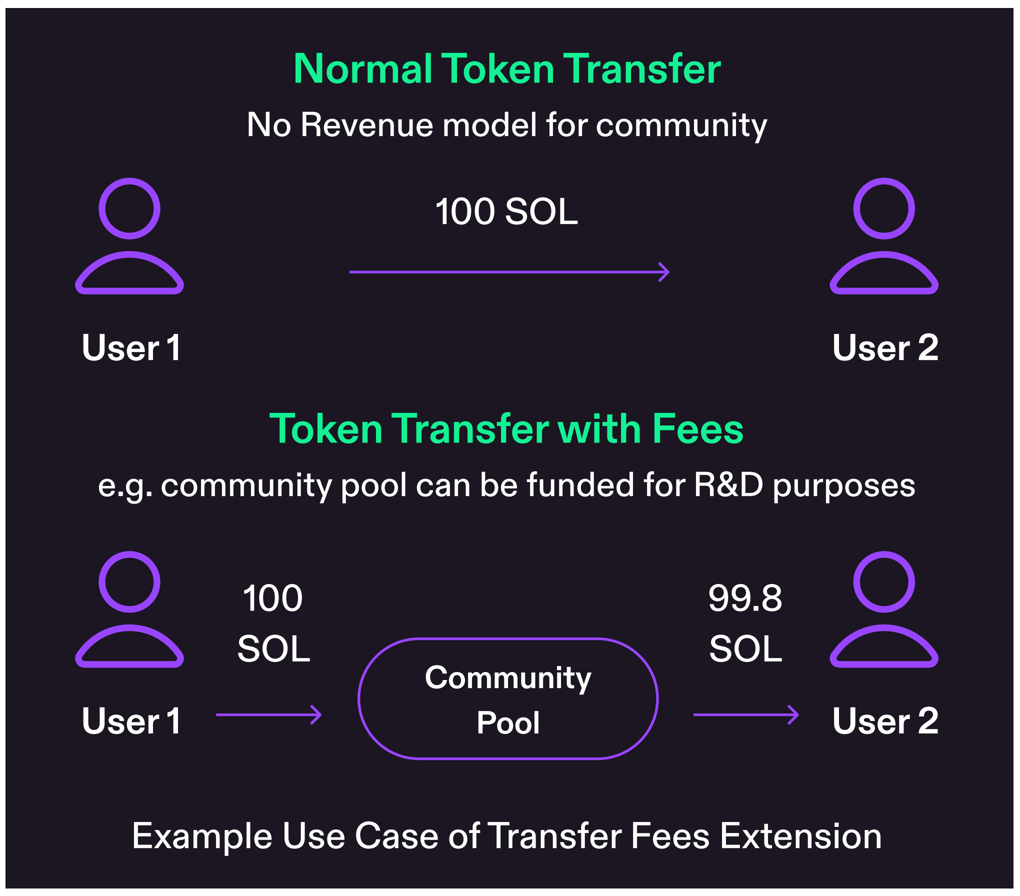
## Transfer Fees

The **transfer fees extension** allows the token minter to assess a fee on every token transfer. In the previous token model, projects that wanted to charge fees typically were required to freeze the account, work through a third party to get the transfer fees, then unfreeze the account.

But with the extension, projects can now create an *automated* transfer of fees, without any additional work. This extension creates an **enforced and embedded** revenue for teams—assessing fees during transfers and custom routing those fees to recipients. Teams can generate royalty payments, platform fees, revenue sharing, and more. DAOs can fund operations, and creators can be paid.



Because the extension is highly configurable, fees can be routed to multiple recipients, capped per transfer, charged on a percentage, and more.


Normal Token Transfer
No Revenue model for community
100 SOL
User 1
User 2
Token Transfer with Fees
e.g. community pool can be funded for R&D purposes
100 SOL
99.8 SOL
User 1
Community Pool
User 2
Example Use Case of Transfer Fees Extension

## Metadata

With the **metadata extension,** modular metadata can be incorporated natively into tokens.

The ability to have native, onchain metadata definitions and storage gives teams a new level of flexibility. Developers can now add, update, and remove custom fields as needed, tailoring the metadata to fit the specific requirements of their projects. With token extensions, **this metadata travels with the token** and is **portable across all apps for seamless operability.**

For example, including modular token-related metadata on-chain lets artists and creators dynamically embed details about their creations. For standard token mints—name, symbol, and URL can easily be directly added to the mint account making it much easier for indexers and protocols to work with your token.

```
$ spl-token update-metadata
5K8RVdjpY3CHujyKjQ7RkyiCJqTG8Kba9krNfpZnmvpS
name YourToken

$ spl-token update-metadata
5K8RVdjpY3CHujyKjQ7RkyiCJqTG8Kba9krNfpZnmvpS
new-field new-value

$ spl-token update-metadata
5K8RVdjpY3CHujyKjQ7RkyiCJqTG8Kba9krNfpZnmvpS
new-field --remove
```

*Updating, adding, and removing custom metadata fields through the CLI*

In a more complex scenario, the extension could be used for supply chain and logistics applications: essential product information (such as origin, date of manufacturing, etc.) could be embedded into the metadata of the tokens representing the products themselves.  Apps can immediately support these tokens, even when they come from other platforms, and even when they have custom attributes.

The metadata schema is highly configurable. With the extension, we anticipate the release of highly innovative applications where detailed, transparent, and modular metadata is crucial.

## Non-transferable Tokens

Sometimes teams don't want their tokens to be traded, exchanged, and transferred; they want the tokens to be non-transferable. This is sometimes referred to as soulbond tokens.

The **non-transferable extension** creates these soulbound tokens—non-fungible tokens minted to a specific address and kept with that original recipient permanently. The tokens become a tamper-proof record of ownership that indelibly links the original minter and the recipient.

Examples include tokens that symbolize personal credentials such as degrees and personal achievements, badges, trophies, or points. They could also be used to give recipients the non-transferable right to vote or make decisions. Soulbound tokens can even be on-chain loyalty cards permanently tied to the user or used as a proof of purchase similar to the recent Saga genesis token given to owners of the Saga mobile phone.

## Interest-earning Tokens

With the **interest-bearing mint extension**, developers can create interest-bearing assets natively onchain. This enables projects to mirror real-world interest-bearing instruments such as bonds, with interest rate configuration, compounded interest, and more.

With the global fixed-income market totaling about $130 trillion, this is a massive use case.

Tokens can have native yield or transparent and automatic rewards without the need for a rebase. This could be useful for DAOs, staked tokens, or debts. With the extension, teams can set an interest rate on their token program and fetch the balance of tokens (with interest) at any time. Interest is continuously compounded based on the network timestamp.

Note that this is only a cosmetic feature. This means that the amount returned is purely for display purposes; no additional tokens are actually minted.

## Immutable Owner

In normal token programs, the token account owner generally has some special rights. These rights can be useful for minting new tokens, temporarily stopping the transfer of tokens, upgrading the program, and taking specific actions in an emergency.

However, token account owners may reassign ownership to another address. Even though this can be useful, it can also be a vulnerability. If the account owner has reassigned ownership of their token account, applications may derive the address for that account and use it, not knowing that it no longer belongs to the owner.

The **immutable owner extension** fixes this issue by creating token accounts that don't allow ownership transfer. Token account ownership is fixed, offering verified integrity and resilient security. With the extension, you can eliminate potential account misappropriation.

## Token Extensions Use Cases by Vertical

With such an extensive range of available token extensions, the list of use cases is extensive and growing. Let's look at a few by vertical to see how your team might take advantage.

### Gaming

In gaming, soulbound, *non-transferable tokens* could be used to uniquely identify players in the form of a UUID. These tokens could also be used to build non-transferable, in-game social profiles.

*Transfer hooks* could be used to ensure everyone who receives a token is an actual player, not a bot. Transfer fees could take a percentage of every transaction on the secondary market to help fund further development of the game. Players could be frozen if they violate rules.

*Confidential transfers* of in-game currencies could ensure privacy for these transactions, allowing players to buy, sell, or trade assets without revealing the transaction amounts to others.

Teams could also use *confidential transfers* and default account state together to gate releases. If a new expansion pack is released, a team could drop tokens for the expansion pack to users but not allow users to know the details or access the tokens until the user pays some fee. Or teams could airdrop tokens to players that don't release until the player reaches a certain level in the game.

### DAOs

Tokens with *immutable owner* and non-transferable could be used for DAO governance decisions. In the future, as blockchain is more widely adopted, decision-making activities could occur onchain in more real-world situations where transparent governance is strongly preferred, such as elections, corporation board meetings, etc.

For funding, *confidential transfers* could be used to keep the amount of funding private. The general public would know who has funded the DAO, but not the amounts.

### DeFi

Tokens that represent real-world assets (RWAs) could combine extensions to create tokens that earn interest, require metadata about their origin, and require KYC with transfer restrictions.

DeFi projects could use *transfer fees* to implement fee-based tokenomics and revenue sharing.

### Stablecoins

With a market already of over $135 billion USD and rising, stablecoins are big business. Token extensions can enable stablecoin teams to build fully compliant stablecoins on Solana.

Using transfer hooks, teams could enforce full KYC on all users or on users who hold over a certain amount. Teams could use permanent delegate (establishes a defined authority that can burn or transfer any amount of tokens) to freeze accounts or remove tokens from an account when fraud or criminal activity occurs.

### Payments

Because of its high performance, scalability, and low cost, Solana is a leading blockchain for payment solutions. In fact, Visa said in a recent deep dive that "Solana's unique technological advantages, including high throughput ... low cost ... and high resiliency ... create a scalable blockchain platform with a compelling value proposition for payments."

Teams creating payment solutions can use extensions to enforce accepted practices and help take their products mainstream, such as including an automatic tax on payments while still keeping the amount of the payment confidential. By adding the memo required, teams could ensure the transfer is documented, tracked, and audited, enabling simplified payment reconciliation.

## Deploying a token with Token Extensions

***How do you migrate existing tokens to token extensions?***
If you are the owner of an existing token with many holders, you can migrate to token extensions by creating a new mint and using the `spl-token-upgrade` program to convert tokens. This program burns old tokens and transfers new ones using an escrow account.

Now that we've seen the benefits of utilizing token extensions, let's look at how simple it is for your team to use the extensions.

Let's assume you want to deploy a token that charges a .25% fee with every transfer. The opt-in feature of token extensions allows you to create an implementation on-chain by simply specifying a tag at the end of your deployment script.

Here's the code:

```
$ spl-token --program-id
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb
create-token --transfer-fee 25 1000000
```

Note:

- the flag `--transfer-fee` that indicates this token will use the transfer fee interface
- the parameter 25 which sets the transfer fee to 25 basis points, or .25% of the tokens transferred
- The parameter 5000 which sets the maximum number of tokens that can be paid as a fee.

That's all it takes! While there are more complex configurations available, at its core token extensions are a straightforward, simple way for your team to enable powerful, enterprise-grade, tokens.

For a more detailed tutorial, please refer to the Solana Documentation.

## Into the Future with Token Extensions

Token extensions are a part of the ever-evolving landscape on Solana. For teams that need flexibility and advanced features in their tokens, token extensions give you a reliable, enterprise-grade solution.

With token extensions you don't need to face the overwhelming complexity of developing, auditing, and deploying custom token contracts. You simply get the capabilities of smart contracts you need with the speed, security, and reliability you require. And with the time and cost saved, you can focus more on the features and business cases truly important to your team.

TO LEARN MORE
solana.com/tokenextensions