IEEE Standard for Blockchain based Digital Asset Identification

Overview

Background

Digital economy has become the consensus of global development. As the core technological element of digital economy, blockchain has developed rapidly from POC verification to small-scale model exploration. The asset identification specification is the key to establishing a digital asset management system, especially when it comes to multi-asset management and cross-chain asset operations. Without a universal digital asset identification specification, asset management based on different protocols will become more and more complicated.Many problems will ensue, such as isolation of technology platform, isolation of single application mode and disconnection of industrial ecology.On the basis of the key standardization objectives of blockchain technology, standards are summarized and best practices are summarized, and a systematic standard family is gradually established with standardized methods, and the rapid and benign development of the industry is guided.

Scope

- Define the data structure related to digital asset identification;
- Define the data types related to digital asset identification;
- Define the data fields related to digital asset identification;
- Define data format specifications related to digital asset identification;
- Propose asset management operation specifications related to digital asset identification.

Purpose

This standard defines the data structure related to asset identification to improve the digital asset management efficiency, provide guidance for the design of digital asset management solutions, and provide a reference for building a digital asset service platform.

Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

Definitions, abbreviations and acronyms

Definitions

For the purposes of this document, the following terms and definitions apply. The IEEE Standards Dictionary Online should be consulted for terms not defined in this clause.

- **Blockchain**: Distributed ledger with confirmed blocks organized in an append-only,sequential chain using cryptographic links.

  NOTE-See[B1]

- **Digital asset**: Asset that exist only in digital form or which is the digital representation of another asset.

  NOTE-See[B1]

- **Data fields**: A data unit described by a set of attributes, including definition, identification, representation and permissible values.

  NOTE-See[B4]

- **Data type**: A format determined by meta-operation of data, and used to collect letters, figures and (or) symbols to describe the value of a data element.

  NOTE-See[B5]

- **Token**: Digital asset that represents a collection of entitlements.

  NOTE-See[B1]

- **Fungible Token**: Fungible Tokens have a property that makes each Token be exactly the same (in type and value) of another Token.

  NOTE-See[B2]

- **Non-Fungible Token**: A Non-Fungible Token (NFT) is used to identify something or someone in a unique way. NOTE-See[B3]

Abbreviations and acronyms

- **FT**: Fungible Token.
- **NFT**: Non-Fungible Token.

Specfication

Methods

Method Name	Request Type	Response Type	Description
Create	CreateInput	Empty	Creating a new token serves as a user's proof of interest. The token can be circulated in the blockchain system with their unique characteristics, such as, Encryption, immutable, uniqueness. And token has certain economic value.
Issue	IssueInput	Empty	Issuing some amount of tokens to an address is the action of increasing that address' balance for the given token. The total amount of issued tokens must not exceed the total supply of the token and only the issuer (creator) of the token can issue tokens. Issuing tokens effectively increases the circulating supply.
Transfer	TransferInput	Empty	Transferring tokens simply is the action of transferring a given amount of tokens from one address to another. The origin or source address is the signer of the transaction. The balance of the sender must be higher than the amount that is transferred.
TransferFrom	TransferFromInput	Empty	The TransferFrom action will transfer a specified amount of tokens from one address to another. For this operation to succeed the from address needs to have approved (see allowances) enough tokens to Sender of this transaction. If successful the amount will be removed from the allowance.
BatchTransferFrom	BatchTransferFromInput	Empty	The BatchTransferFrom action will transfer a batch of specified amount of tokens from one address to another. For this operation to succeed the from address needs to have approved (see allowances) enough tokens to Sender of this transaction. If successful the amount will be removed from the allowance.
Approve	ApproveInput	Empty	The approve action increases the allowance from the Sender to the Spender address, enabling the Spender to call TransferFrom and approve a certain amount of transactions and to extract a certain amount of money.
UnApprove	UnApproveInput	Empty	This is the reverse operation for Approve, it will decrease the allowance and revoke permission from operation of Approve.

Method Name	Request Type	Response Type	Description
Lock	LockInput	Empty	This method is used to lock tokens with specific symbol.
Unlock	UnlockInput	Empty	This is the reverse operation of Lock, which un-locks some previously locked tokens.
Burn	BurnInput	Empty	This action will burn the specified amount of tokens which are burnable, and remove them from the token's Supply.
ChangeTokenIssuer	ChangeTokenIssuerInput	Empty	Change the issuer of the specified token. Only the original issuer can change it.
CrossChainTransfer	CrossChainTransferInput	Empty	This interface is a cross-chain transfer transation.
CrossChainReceiveToken	CrossChainReceiveTokenInput	Empty	This method is used to receive cross-chain transfer transactions.
CrossChainCreateToken	CrossChainCreateTokenInput	Empty	This method is to deal with the info of tokens created by other chains, which exist in transfer transactions.
CrossChainTransferBatch	CrossChainTransferBatchInput	Empty	This interface is used for batch cross-chain transfer.
GetTokenInfo	GetTokenInfoInput	TokenInfo	View method to query token information.
GetBalance	GetBalanceInput	GetBalanceOutput	View method to query the balance at the specified address.
GetBalanceBatch	GetBalanceBatchInput	GetBalanceBatchOutput	View method to batch query the balance at the specified address.
GetAllowance	GetAllowanceInput	GetAllowanceOutput	View method to query the account's allowance for other addresses
GetLockedAmount	GetLockedAmountInput	GetLockedAmountOutput	View method to query the information for a lock.
GetCrossChainTransferTokenContractAddress	GetCrossChainTransferTokenContractAddressInput	Address	View method to query the address of receiving token in cross-chain transfer.

Data structures

- **Address** The function of address return. Output parameters: address.

Field	Type	Label	Description
address	string		Address.

- **ApproveForAllInput** The function of increasing the allowance of all one's tokens from the Sender to the Spender address, Input parameters: spender, symbol.

Field	Type	Label	Description
spender	Address		The address of an account/contract that is approved to make the operate.

Field	Type	Label	Description
symbol	string		The symbol of token to approve.

- **ApproveInput** The function of increasing the allowance from the Sender to the Spender address, Input parameters: spender, symbol, amount.

Field	Type	Label	Description
spender	Address		The address that allowance will be increased.
symbol	string		The symbol of token to approve.
amount	int64		The amount of token to approve.

- **Approved** The event of increasing the allowance from the Sender to the Spender address. Input parameters: owner, spender, symbol, amount.

Field	Type	Label	Description
owner	Address		The address of the token owner.
spender	Address		The address that allowance be increased.
symbol	string		The symbol of approved token.
amount	int64		The amount of approved token.

- **BatchTransferFromInput** The function of batch transfering form. Input parameters: from, to, symbol, amount, memo.

Field	Type	Label	Description
from	Address		The source address of the token.
to	Address		The destination address of the token.
symbol	string	repeated	The symbol of the token to transfer.
amount	int64	repeated	The amount to transfer.
memo	string		The memo.

- **BoolValue** The function of BoolValue return. Output parameters: bool_value.

Field	Type	Label	Description
bool_value	bool		Bool value.

- **BurnInput** The function of burning the specified amount of tokens, removing them from the token's Supply. Input parameters: symbol, amount, memo.

Field	Type	Label	Description
symbol	string		The symbol of token to burn.
amount	int64		The amount of token to burn.

- **Burned** The event of burning the specified amount of tokens, removing them from the token's Supply. Input parameters: burner, symbol, amount.

Field	Type	Label	Description
burner	Address		The address who wants to burn token.
symbol	string		The symbol of burned token.
amount	int64		The amount of burned token.

- **ChangeTokenIssuerInput** The function of changing the issuer of the specified token. Input parameters: symbol, new_token_issuer.

Field	Type	Label	Description
symbol	string		The token symbol.
new_token_issuer	Address		The new token issuer for change.

- **CreateInput** The function of creating a token. Input parameters: symbol, token_name, supply, total_supply, decimals, issuer, is_burnable, issue_chain_id, issued, external_information.

Field	Type	Label	Description

Field	Type	Label	Description
symbol	string		The symbol of the token.
token_name	string		The full name of the token.
total_supply	int64		The total supply of the token.
decimals	int32		Precision of token. When decimal =0, the token created belongs to Non-Fungible Token otherwise, it belongs to Fungible Token.
issuer	Address		The address that created the token.
is_burnable	bool		A flag indicating if this token is burnable.
exter_info	external_information		The external information aimed to different kinds of token. Fields starting with a double underscore are standard custom fields. Other fields can be customized by the developer. for example, if the token belongs to Non-Fungible Token, it uses ERC-721 protocal, and it has its own identity, if the token belongs to Fungible Token, it uses ERC-20protocal. Showing like car->porsche101.

- **CrossChainBatchReceived** The event of batch receiving the token(cross-chain). Input parameters: from, to, symbol, amount, from_chain_id, issue_chain_id, parent_chain_height.

Field	Type	Label	Description
from	Address		The source address of the transferred token.
to	Address		The destination address of the transferred token.
symbol	string	repeated	The symbol of the received token.
amount	int64	repeated	The amount of the received token.
from_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.
parent_chain_height	int64		The parent chain height of the transfer transaction.

- **CrossChainCreateTokenInput** The function of creating tokens on side chain. Input parameters: from_chain_id, parent_chain_height, transaction_bytes, merkle_path.

Field	Type	Label	Description
from_chain_id	int32		The chain id of the chain on which the token was created.
parent_chain_height	int64		The height of the transaction that created the token.
transaction_bytes	bytes		The transaction that created the token.
merkle_path	MerklePath		The merkle path created from the transaction that created the transaction.

- **CrossChainReceiveTokenInput** The function of receiving cross-chain transfers. Input parameters: from_chain_id, parent_chain_height, transfer_transaction_bytes, merkle_path

Field	Type	Label	Description
from_chain_id	int32		The source chain id.
parent_chain_height	int64		The height of the transfer transaction.
transfer_transaction_bytes	bytes		The raw bytes of the transfer transaction.
merkle_path	MerklePath		The merkle path created from the transfer transaction.

- **CrossChainReceived** The event of receiving the token(cross-chain). Input parameters: from, to, symbol, amount, memo, from_chain_id, issue_chain_id, parent_chain_height.

Field	Type	Label	Description
from	Address		The source address of the transferred token.
to	Address		The destination address of the transferred token.
symbol	string		The symbol of the received token.
amount	int64		The amount of the received token.
memo	string		The memo.

Field	Type	Label	Description
from_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.
parent_chain_height	int64		The parent chain height of the transfer transaction.

- **CrossChainTransferBatchInput** The function of batch transfering(cross-chain). Input parameters: from, to, symbol, amount, to_chain_id, issue_chain_id, memo.

Field	Type	Label	Description
from	Address		The signer of the transaction.
to	Address		The receiver of transfer.
symbol	string	repeated	The symbol of token.
amount	int64	repeated	The amount of token to transfer.
to_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.
memo	string		The memo.

- **CrossChainTransferInput** The function of cross-chain transfering. Input parameters: to, symbol, amount, memo, to_chain_id, issue_chain_id.

Field	Type	Label	Description
to	Address		The receiver of transfer.
symbol	string		The symbol of token.
amount	int64		The amount of token to transfer.
memo	string		The memo.
to_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.

- **CrossChainTransferred** The event of transfering tokens(cross-chain). Input parameters: from, to, symbol, amount, to_chain_id, issue_chain_id, memo.

Field	Type	Label	Description
from	Address		The source address of the transferred token.
to	Address		The destination address of the transferred token.
symbol	string		The symbol of the transferred token.
amount	int64		The amount of the transferred token.
to_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.
memo	string		The memo.

- **CrossChainTransferredBatch** The event of batch transfering the token(cross-chain). Input parameters: from, to, symbol, amount, to_issue_chain_id, issue_chain_id.

Field	Type	Label	Description
from	Address		The source address of the transferred token.
to	Address		The destination address of the transferred token.
symbol	string	repeated	The symbol of the transferred token.
amount	int64	repeated	The amount of the transferred token.
to_chain_id	int32		The destination chain id.
issue_chain_id	int32		The chain id of the token.

- **Empty** The function of empty return. Output parameters: empty.

Field	Type	Label	Description
empty	string		Empty return.

- **GetAllowanceInput** The funciton of querying the account's allowance for other addresses. Input parameters: symbol, owner, spender.

Field	Type	Label	Description
symbol	string		The symbol of token.

Field	Type	Label	Description
owner	Address		The address of the token owner.
spender	Address		The address of the spender.

- **GetAllowanceOutput** The output of querying the account's allowance for other addresses. Out parameters: symbol, owner, spender, allowance.

Field	Type	Label	Description
symbol	string		The symbol of token.
owner	Address		The address of the token owner.
spender	Address		The address of the spender.
allowance	int64		The amount of allowance.

- **GetBalanceBatchInput** The function of batch querying the balance at the specified address. Input parameters: symbol, owner.

Field	Type	Label	Description
symbol	string	repeated	The symbol of token.
owner	Address		The target address of the query.

- **GetBalanceBatchOutput** The output of batch querying the balance at the specified address. Output parameters: symbol, owner, balance.

Field	Type	Label	Description
symbol	string	repeated	The symbol of token.
owner	Address		The target address of the query.
balance	int64	repeated	The balance of the owner.

- **GetBalanceInput** The function of querying the balance at the specified address. Input parameters: symbol, owner.

Field	Type	Label	Description
symbol	string		The symbol of token.
owner	Address		The target address of the query.

- **GetBalanceOutput** The output of querying the balance at the specified address. Output parameters: symbol, owner, balance.

Field	Type	Label	Description
symbol	string		The symbol of token.
owner	Address		The target address of the query.
balance	int64		The balance of the owner.

- **GetCrossChainTransferTokenContractAddressInput** The function of querying the address of receiving token in cross-chain transfer. Input parameters: chain_id.

Field	Type	Label	Description
chainId	int32		The chain id.

- **GetLockedAmountInput** The function of querying the information for a lock. Input parameters: address, symbol, lock_id.

Field	Type	Label	Description
address	Address		The address of the lock.
symbol	string		The token symbol.
lock_id	Hash		The id of the lock.

- **GetLockedAmountOutput** The output of querying the information for a lock. Input parameters: address, symbol, lock_id, amount.

Field	Type	Label	Description
address	Address		The address of the lock.

Field	Type	Label	Description
symbol	string		The token symbol.
lock_id	Hash		The id of the lock.
amount	int64		The locked amount.

- **GetTokenInfoInput** The function of querying token information. Input parameters: symbol.

Field	Type	Label	Description
symbol	string		The symbol of token.

- **Hash** The function of hash return. Output parameters: hash.

Field	Type	Label	Description
hash	string		Hash value.

- **Int32Value** The function of Int32Value return. Output parameters: int_value.

Field	Type	Label	Description
int_value	int32		Int32 value.

- **IssueInput** The function of issuing some amount of tokens to an address. Input parameters: symbol, amount, to, memo.

Field	Type	Label	Description
symbol	string		The token symbol to issue.
amount	int64		The token amount to issue.
memo	string		The memo.
to	Address		The target address to issue.

- **Issued** The event of issuing token. Input parameters: symbol, amount, memo, to.

Field	Type	Label	Description
symbol	string		The symbol of issued token.
amount	int64		The amount of issued token.
memo	string		The memo.
to	Address		The issued target address.

- **LockInput** The function of locking tokens. Input parameters: lock_address, lock_id, symbol, amount, memo.

Field	Type	Label	Description
address	Address		The one want to lock his token.
lock_id	Hash		Id of the lock.
symbol	string		The symbol of the token to lock.
memo	string		a memo.
amount	int64		The amount of tokens to lock.

- **MerklePath** The function of merklePath return. Output parameters: merkle_path.

Field	Type	Label	Description
merkle_path	string		Merkle path.

- **StringValue** The function of StringValue return. Output parameters: string_value.

Field	Type	Label	Description
string_value	string		String value.

- **TokenCreated** The event of creating token. Input parameters: symbol, token_name, total_supply, decimals, issuer, is_burnable, issue_chain_id, external information..

Field	Type	Label	Description
symbol	string		The symbol of the token.
token_name	string		The full name of the token.
total_supply	int64		The total supply of the token.
decimals	int32		Precision of token. When decimal =0, the token created belongs to Non-Fungible Token otherwise, it belongs to Fungible Token.
issuer	Address		The address that created the token.
is_burnable	bool		A flag indicating if this token is burnable.
issue_chain_id	int32		The chain id of the token.
exter_info	external_information		The external information aimed to different kinds of token. Fields starting with a double underscore are standard custom fields. Other fields can be customized by the developer. for example, if the token belongs to Non-Fungible Token, it uses ERC-721 protocal, and it has its own identity, if the token belongs to Fungible Token, it uses ERC-20protocal. Showing like car->porsche101.

- **TokenInfo** The information of token. Output parameters: symbol, token_name, supply, total_supply, decimals, issuer, is_burnable, issue_chain_id, issued, external_information.

Field	Type	Label	Description
symbol	string		The symbol of the token.
token_name	string		The full name of the token.
supply	int64		The current supply of the token.
total_supply	int64		The total supply of the token.
decimals	int32		Precision of token. When decimal =0, the token created belongs to Non-Fungible Token, otherwise, it belongs to Fungible Token.
issuer	Address		The address that created the token.

Field	Type	Label	Description
is_burnable	bool		A flag indicating if this token is burnable.
issue_chain_id	int32		The chain id of the token.
issued	int64		The amount of issued tokens.
exter_info	external_information		The external information aimed to different kinds of token. Fields starting with a double underscore are standard custom fields. Other fields can be customized by the developer. for example, if the token belongs to Non-Fungible Token, it uses ERC-721 protocal, and it has its own identity, if the token belongs to Fungible Token, it uses ERC-20protocal. Showing like car->porsche101.

- **TokenInfoList** The output of tokeninfo(list). Output parameters: value

Field	Type	Label	Description
value	TokenInfo	repeated	List of token information.

- **TotalSupply** The function of querying the total supply of the token. Input parameters: owner, symbol, amount, memo.

Field	Type	Label	Description
owner	Address		The owner who issued the token.
symbol	string		The symbol of the transferred token.
amount	int64		The amount of the transferred token.
memo	string		The memo.

- **TransferFromInput** The function of transfering a specified amount of tokens from one address to another. Input parameters: from, to, symbol, amount, memo.

Field	Type	Label	Description

Field	Type	Label	Description
from	Address		The source address of the token.
to	Address		The destination address of the token.
symbol	string		The symbol of the token to transfer.
amount	int64		The amount to transfer.
memo	string		The memo.

- **TransferInput** The function of transferring a given amount of tokens from one address to another. Input parameters: to, symbol, amount, memo.

Field	Type	Label	Description
to	Address		The receiver of the token.
symbol	string		The token symbol to transfer.
amount	int64		The amount to to transfer.
memo	string		The memo.

- **Transferred** The event of transfering tokens. Input parameters: from, to, symbol, amount, memo.

Field	Type	Label	Description
from	Address		The source address of the transferred token.
to	Address		The destination address of the transferred token.
symbol	string		The symbol of the transferred token.
amount	int64		The amount of the transferred token.
memo	string		The memo.

- **TransferredBatch** The event of transfering the token. Input parameters: from, to, symbol, amount.

Field	Type	Label	Description
from	Address		The source address of the transferred token.

Field	Type	Label	Description
to	Address		The destination address of the transferred token.
symbol	string	repeated	The symbol of the transferred token.
amount	int64	repeated	The amount of the transferred token.

- **URI** The event of showing infomation of the changed token. Input parameters: symbol, amount.

Field	Type	Label	Description
symbol	string		The symbol of issued token.
amount	int64		The amount of issued token.

- **UnApproveInput** The function of reversing operation for Approve, it will decrease the allowance. Input parameters: spender, symbol, amount.

Field	Type	Label	Description
spender	Address		The address that allowance will be decreased.
symbol	string		The symbol of token to un-approve.
amount	int64		The amount of token to un-approve.

- **UnApproved** The event of reversing operation for Approve, it will decrease the allowance. Input parameters: owner, spender, symbol, amount.

Field	Type	Label	Description
owner	Address		The address of the token owner.
spender	Address		The address that allowance be decreased.
symbol	string		The symbol of un-approved token.
amount	int64		The amount of un-approved token.

- **UnlockInput** The function of unlocking tokens. Input parameters: unlock_address, lock_id, symbol, amount, memo.

Field	Type	Label	Description
address	Address		The one want to un-lock his token.
lock_id	Hash		Id of the lock.
symbol	string		The symbol of the token to un-lock.
memo	string		a memo.
amount	int64		The amount of tokens to un-lock.

- **external_information** the extra information of token, Including standard preset fields and other fields that can be customized by the developers. Output parameters: description, image, properties, and other fields.

Field	Type	Label	Description
__description	string		Describes the asset to which this token represents.
__image	string		A URI pointing to a resource with mime type image/* representing the asset to which this token represents. Consider making any images at a width between 320 and 1080 pixels and aspect ratio between 1.91:1 and 4:5 inclusive.
__properties	string	repeated	Arbitrary set of attributes.
user_define	external_information.UserDefineEntry	repeated	Aimed at Non-Fungible Token, which has its own identification. showing like car->porsche101.

- **external_information.UserDefineEntry**

Field	Type	Label	Description
key	string		
value	string		

Others

- **Scalar Value Types**

.proto Type	Notes	C++	Java	Python	Go	C#	PHP	Ruby
double		double	double	float	float64	double	float	Float
float		float	float	float	float32	float	float	Float
int32	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint32 instead.	int32	int	int	int32	int	integer	Bignum or Fixnum (as required)
int64	Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint64 instead.	int64	long	int/long	int64	long	integer/string	Bignum
uint32	Uses variable-length encoding.	uint32	int	int/long	uint32	uint	integer	Bignum or Fixnum (as required)
uint64	Uses variable-length encoding.	uint64	long	int/long	uint64	ulong	integer/string	Bignum or Fixnum (as required)

.proto Type	Notes	C++	Java	Python	Go	C#	PHP	Ruby
sint32	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int32s.	int32	int	int	int32	int	integer	Bignum or Fixnum (as required)
sint64	Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int64s.	int64	long	int/long	int64	long	integer/string	Bignum
fixed32	Always four bytes. More efficient than uint32 if values are often greater than 2^28.	uint32	int	int	uint32	uint	integer	Bignum or Fixnum (as required)
fixed64	Always eight bytes. More efficient than uint64 if values are often greater than 2^56.	uint64	long	int/long	uint64	ulong	integer/string	Bignum
sfixed32	Always four bytes.	int32	int	int	int32	int	integer	Bignum or Fixnum (as required)
sfixed64	Always eight bytes.	int64	long	int/long	int64	long	integer/string	Bignum
bool		bool	boolean	boolean	bool	bool	boolean	TrueClass/FalseClass
string	A string must always contain UTF-8 encoded or 7-bit ASCII text.	string	String	str/unicode	string	string	string	String (UTF-8)
bytes	May contain any arbitrary sequence of bytes.	string	ByteString	str	[]byte	ByteString	string	String (ASCII-8BIT)

Bibliography

Bibliographical references are resources that provide additional or helpful material but do not need to be understood or used to implement this standard. Reference to these resources is made for informational use only.

[B1] ISO 22739:2020(en), Blockchain and distributed ledger technologies - Vocabulary.

[B2] ERC-20 Token Standard.

[B3] ERC-721 Token Standard.

[B4] GB/T 19488.1-2004.

[B5] GB/T 18391.1-2002.