Contract Changes Documentation - June 2025 Update

This document outlines the key changes between the old and new contract implementations for the Alvara Protocol. These changes affect both frontend and backend integrations, and this guide will help teams update their code accordingly.

Major Terminology Changes

1. BTS to BSKT Renaming

All references to "BTS" (Basket Token Standard) have been renamed to "BSKT" throughout the codebase:

Contract Names:

- `BasketTokenStandard.sol` remains the same, but all references use BSKT instead of BTS
- `BTSPair.sol` → `BSKTPair.sol`
- Interface files: `IBTS.sol` → `IBSKT.sol`, `IBTSPair.sol` → `IBSKTPair.sol`

Variable and Function Names:

- `btsImplementation` → `bsktImplementation`
- `btsPairImplementation` → `bsktPairImplementation`
- `listOfBTS` → `bsktList`
- `getBTSAtIndex` → `getBSKTAtIndex`
- `totalBTS` → `totalBSKT`
- `btsPair` → `bsktPair`

Event Names:

- `CreatedBTS` → `BSKTCreated`
- `UpdatedBTSImplementation` → `BSKTImplementationUpdated`
- `UpdatedBTSPairImplementation` → `BSKTPairImplementationUpdated`
- `BTSCreationFeeDeducted` → `BSKTCreationFeeDeducted`
- `ContributedToBTS` → `ContributedToBSKT`
- `WithdrawnFromBTS` → `WithdrawnFromBSKT`
- `WithdrawnETHFromBTS` → `WithdrawnETHFromBSKT`
- `RebalanceBTS` → `BSKTRebalanced`

Error Names:

- `InsufficientBTSCreationAmount` → `InsufficientBSKTCreationAmount`
- `InvalidBTSImplementation` → `InvalidBSKTImplementation`
- `InvalidBTSPairImplementation` → `InvalidBSKTPairImplementation`

2. Slippage to Buffer Renaming

All references to "slippage" have been renamed to "buffer" throughout the codebase:

Function Parameters:

- `_slippage` → `_buffer` in all function signatures

Error Names:

- `InvalidSlippage` → `InvalidBuffer`

Event Parameters:

- `_slippage` → `_buffer` in event parameters

Contract Architecture Changes

1. Access Control System

Old Contract: Used `OwnableUpgradeable` with a single owner having all permissions.

```
contract Factory is Initializable, OwnableUpgradeable,
ReentrancyGuardUpgradeable {
```

New Contract: Implements `AccessControlUpgradeable` with role-based permissions.

```
contract Factory is Initializable, AccessControlUpgradeable,
ReentrancyGuardUpgradeable, IFactory {
```

Integration Impact:

- Frontend: Update admin panels to reflect different permission levels
- Backend: Use role-checking functions instead of `onlyOwner` modifier
- API: Update authentication logic to account for different roles

2. Role Definitions

New Contract Only: Defines specific roles for different administrative functions:

```
bytes32 public constant ADMIN_ROLE = keccak256("ADMIN_ROLE");
bytes32 public constant FEE_MANAGER_ROLE = keccak256("FEE_MANAGER_ROLE");
bytes32 public constant URI_MANAGER_ROLE = keccak256("URI_MANAGER_ROLE");
bytes32 public constant WHITELIST_MANAGER_ROLE =
keccak256("WHITELIST_MANAGER_ROLE");
bytes32 public constant UPGRADER_ROLE = keccak256("UPGRADER_ROLE");
```

3. Interface Implementation

Old BSKT Contract:

```
contract BasketTokenStandard is
    ERC721URIStorageUpgradeable,
    IERC2981Upgradeable,
    ReentrancyGuardUpgradeable
```

New BSKT Contract: Added IBSKT interface implementation

```
contract BasketTokenStandard is
    ERC721URIStorageUpgradeable,
    IERC2981Upgradeable,
    ReentrancyGuardUpgradeable,
    IBSKT
```

4. Library Usage

Old Contracts: Used custom code for contract detection

New Contracts: Added `AddressUpgradeable` library for more reliable contract detection

```
import "@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol";
```

State Variable Changes

1. Type Changes

Old Factory Contract:

```
struct PlatformFeeConfig {
    uint256 btsCreationFee;
    uint256 contributionFee;
    uint256 withdrawalFee;
    address feeCollector;
}

uint public minPercentALVA;
address public ROUTER;
address public WETH;
address[] public listOfBTS;
uint public monthlyFee;
```

New Factory Contract:

```
struct PlatformFeeConfig {
    uint16 bsktCreationFee;
    uint16 contributionFee;
    uint16 withdrawalFee;
    address feeCollector;
}

uint16 public minPercentALVA;
address public router;
address public weth;
address[] public bsktList;
uint256 public monthlyFeeRate;
uint256 private _minLpWithdrawal;
```

Integration Impact:

- Data Types: Update any code that reads these variables to handle the new types
- Variable Names: Update references from `ROUTER` to `router`, `WETH` to `weth`, `listOfBTS` to `bsktList`
- Fee Handling: `uint16` has a smaller range than `uint256`, ensure values fit within range

2. New Variables

New Factory Contract Only:

```
uint256 private _minLpWithdrawal; // Minimum LP tokens required for
withdrawal
```

Integration Impact:

- Frontend: Add UI elements for minimum LP withdrawal settings
- Backend: Check against this minimum when processing withdrawals

3. Removed Variables

Old BSKT Contract:

```
mapping(address => bool) private _isTokenPresent;
```

New BSKT Contract: This mapping has been removed, and token presence is now checked differently.

Function Changes

1. Initialization

Old Factory Contract:

```
function initialize(...) external initializer {
    __Ownable_init();
    __ReentrancyGuard_init();
    // ...
}
```

New Factory Contract:

```
function initialize(...) external initializer {
    _grantRole(DEFAULT_ADMIN_ROLE, _msgSender());
    __ReentrancyGuard_init();
    // Additional validation for collectionUri
    // ...
    _minLpWithdrawal = 1e11; // Set default minimum LP amount
}
```

2. Create Function

Old Factory Contract:

```
function createBTS(
    // ...parameters
    uint _slippage
) external payable nonReentrant {
    // Implementation
}
```

New Factory Contract:

```
function createBSKT(
    // ...parameters
    uint256 _buffer,
    uint256 _deadline
) external payable nonReentrant {
    // Implementation with deadline check and additional validations
    if (block.timestamp > _deadline) revert DeadlineInPast(_deadline);
}
```

3. Administrative Functions

Old Factory Contract: Used `onlyOwner` modifier

```
function updateBTSImplementation(address _btsImplementation) external
onlyOwner {
    // Implementation
}
```

New Factory Contract: Uses role-based access control

```
function updateBSKTImplementation(address _bsktImplementation) external
onlyRole(UPGRADER_ROLE) {
    // Implementation with additional contract validation
}
```

4. New Functions

New Factory Contract Only:

```
function minLpWithdrawal() external view returns (uint256)
function setMinLpWithdrawal(uint256 newMin) external onlyRole(ADMIN_ROLE)
```

5. Renamed Functions

Old Factory Contract:

```
function removeWhitelistedContract(address contractAddr) external onlyOwner
```

New Factory Contract:

```
function dewhitelistContract(address contractAddr) external
onlyRole(WHITELIST_MANAGER_ROLE)
```

6. Modified BSKT Modifiers

Old BSKT Contract:

```
modifier checkLength(uint lengthOne, uint lengthTwo) {
    if (lengthOne != lengthTwo) revert InvalidLength();
    _;
}
```

New BSKT Contract:

```
modifier checkLength(uint256 lengthOne, uint256 lengthTwo) {
    if (lengthOne != lengthTwo || lengthOne == 0 || lengthTwo == 0)
        revert InvalidLength();
    _;
}
```

New BSKT Contract Only:

```
modifier validateMinLpWithdrawal(uint256 amount) {
    uint256 min = _factory().minLpWithdrawal();
    if (amount < min) revert InvalidWithdrawalAmount();
    _;
}
```

Event Changes

1. Renamed Events

Old Factory Contract:

```
event CreatedBTS(..., uint _slippage, ...);
event UpdatedAlva(address indexed alva);
event UpdatedMinPercentALVA(uint256 indexed percent);
event UpdatedBTSImplementation(address indexed btsImplementation);
event UpdatedBTSPairImplementation(address indexed btsPairImplementation);
event UpdatedMonthlyFee(uint256 newMonthlyFee);
```

New Factory Contract:

```
event BSKTCreated(..., uint256 _buffer, ...);
event AlvaUpdated(address alva);
event MinAlvaPercentageUpdated(uint256 percent);
event BSKTImplementationUpdated(address indexed bsktImplementation);
event BSKTPairImplementationUpdated(address indexed
bsktPairImplementation);
// No direct equivalent for UpdatedMonthlyFee
```

2. New Events

New Factory Contract Only:

```
event MinBSKTCreationAmountUpdated(address indexed caller, uint256 amount);
event ContractWhitelisted(address indexed contractAddress);
event ContractRemovedFromWhitelist(address indexed contractAddress);
event MinLpWithdrawalUpdated(uint256 newMinLpWithdrawal);
```

Error Handling Changes

1. Renamed Errors

Old Factory Contract:

```
error InvalidSlippage(uint provided, uint minAllowed, uint maxAllowed);
error RoyaltyUnchanged();
error InvalidFeeAmount(uint256 deductedFee);
error InvalidMgmtFee(uint value, uint maxAllowed);
```

New Factory Contract:

```
error InvalidBuffer(uint256 provided, uint256 minAllowed, uint256
maxAllowed);
error DuplicateRoyaltyValue();
// No direct equivalent for InvalidFeeAmount
// No direct equivalent for InvalidMgmtFee
```

2. New Errors

New Factory Contract Only:

```
error DeadlineInPast(uint256 deadline);
error InvalidTokensAndWeights();
error InvalidWithdrawalAmount();
```

Calculation Changes

1. Management Fee Calculation

Old Factory Contract:

```
function calMgmtFee(uint256 months, uint256 lpSupply) external view returns
(uint256) {
    // ...calculation
    uint256 denominator = 1e18 - (1e18 - powerValue);
    uint256 lpAmount = numerator / denominator;
    return lpAmount;
}
```

New Factory Contract:

```
function calMgmtFee(uint256 months, uint256 lpSupply) external view returns
(uint256) {
    // ...calculation
    uint256 lpFeeAmount = numerator / powerValue;
    return lpFeeAmount;
}
```

Implementation Improvements

1. Contract Validation

New Factory Contract Only: Adds contract validation using `AddressUpgradeable.isContract()`

```
if (_bsktImplementation == address(0) ||
!AddressUpgradeable.isContract(_bsktImplementation))
    revert InvalidAddress();
```

2. Token Amount Tracking

Old Factory Contract: Did not track actual received token amounts

New Factory Contract: Tracks token amounts received through balance differentials

```
uint256 balance = IERC20(_tokens[i]).balanceOf(_bsktPair);
// ...swap execution...
amounts[i] = IERC20(_tokens[i]).balanceOf(_bsktPair) - balance;
```

3. Deadline Validation

New Contract Only: Adds deadline validation to prevent transaction execution after expiry

```
if (block.timestamp > _deadline) revert DeadlineInPast(_deadline);
```

Integration Impact Summary

Frontend Integration

1. Update function calls:
   - Rename `slippage` parameter to `buffer`
   - Add `deadline` parameter to createBSKT calls
   - Use role-specific functions for admin actions
2. Update UI components:
   - Add role-based permission controls
   - Add minimum LP withdrawal settings

- Update error messages to reflect new error types

3. Update event listeners:

    - Listen for renamed events (e.g., `BSKTCreated` instead of `CreatedBTS`)
    - Add listeners for new events

Backend Integration

1. Update contract interaction:

    - Use role-based access control functions
    - Include deadline parameter in transactions
    - Rename variable references (ROUTER → router, WETH → weth)

2. Update data handling:

    - Handle uint16 fee values instead of uint256
    - Check against minimum LP withdrawal amounts
    - Update management fee calculation logic

3. Update security measures:

    - Implement role-based permissions in backend services
    - Validate contract addresses where appropriate

API Changes

1. Update endpoints that interact with Factory contract:

    - Add deadline parameter to creation endpoints
    - Rename slippage to buffer in request/response schemas
    - Add role-based authentication

2. Update response handling:

    - Handle new error types
    - Process renamed events
    - Track new events for analytics